
**SPC57EM80xx - 32-bit Power Architecture® based MCU with up to
4 Mbyte Flash and 304 Kbyte RAM memories**

Preface

Overview

The primary objective of this document is to define the functionality of the SPC57EM80xx microcontroller for use by software and hardware developers. The SPC57EM80xx is built on Power Architecture® technology and integrates technologies that are important for today's and tomorrow's automotive applications. SPC57EM80xx applications include automotive powertrain controllers for six to eight cylinder gasoline and diesel engines, transmission control, chassis control, steering and braking, as well as high end hybrid and advanced combustion systems, electric power steering, and safety applications that require a high level of safety integrity.

The information in this book is subject to change without notice, as described in the disclaimers on the title page. As with any technical documentation, it is the reader's responsibility to be sure he or she is using the most recent version of the documentation.

To locate any published errata or updates for this document, visit the ST Web site at www.st.com.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the SPC57EM80xx device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

Document organization

This document includes chapters that are divided into parts:

- Part I includes chapters that describe the device as a whole or provide device-specific information.
- Part II includes chapters that describe the functionality of the individual modules on the device.
- Part III includes chapters that describe the functionality of the Emulation and Debug Device (ED).

Contents

1	Register conventions	154
1.1	Acronyms and abbreviations	155
1.2	References	156
2	Introduction	157
2.1	SPC57EM80xx microcontroller	157
2.1.1	Target applications	157
2.1.2	Overall architecture	157
2.1.3	Core features	158
2.1.4	I/O Processor	158
2.1.5	Memory hierarchy	159
2.2	Features summary	160
2.3	Feature list	161
2.4	Packaging	163
2.5	Software debug and calibration	163
2.6	Block diagrams	164
3	Embedded memories	166
3.1	Overview	166
3.2	SRAM	166
3.2.1	System SRAM	166
3.2.2	Standby SRAM Regulator	166
3.2.3	Overlay SRAM	168
3.2.4	Processor core local SRAM	169
3.3	Embedded flash memory	169
3.3.1	Flash memory controller	170
3.3.2	Flash memory array	170
3.4	End-to-end Error Correction Code (e2eECC)	172
3.5	Security features	174
4	Signal Description	176
4.1	Production packages	176
4.2	Emulation package	176

4.3	Package pinouts and ballouts	176
4.4	Pin/ball descriptions	176
4.4.1	Pin/ball startup and reset states	177
4.4.2	Power supply and reference voltage pins/balls	177
4.4.3	System pins/balls	179
4.4.4	LVDS pins/balls	179
4.4.5	Generic pins/balls	184
5	Memory Map	185
6	Functional Safety	202
6.1	Introduction	202
6.2	Safety overview	202
6.3	Module categorization	202
6.4	System implementation	205
6.4.1	General concept	205
6.4.2	Dual-core lockstep	226
6.4.3	Common Cause Failure measures	229
6.4.4	ECC	229
7	Device Configuration	233
7.1	Introduction	233
7.2	Core modules	233
7.2.1	Core Reset Settings	233
7.2.2	Special Purpose Register Summary	236
7.3	System modules	238
7.3.1	SIUL2 configuration	238
7.3.2	Semaphores2 (SEMA42)	238
7.3.3	Crossbar switch configuration	239
7.3.4	System Memory Protection Unit (SMPU) configuration	243
7.3.5	Peripheral bridge configuration	248
7.3.6	Platform Configuration Module (PCM)	248
7.3.7	Interrupt Controller (INTC) configuration	253
7.3.8	DMA Controller configuration	295
7.3.9	DMACHMUX configuration	295
7.4	Clocking	300

7.5	Memories and memory interfaces	301
7.5.1	Flash memory controller (PFLASH) configuration	301
7.6	Analog modules	303
7.6.1	Temperature sensor configuration	304
7.6.2	Analog to Digital Converters (ADC) configuration	304
7.7	Timers	305
7.7.1	System Timer Module (STM) configuration	305
7.7.2	Software Watchdog Timer (SWT) configuration	306
7.7.3	PIT configuration	308
7.7.4	GTM 103 Integration Module (GTMINT103) configuration	310
7.8	Communication interfaces	311
7.8.1	Fast Ethernet Controller (FEC) configuration	311
7.8.2	FEC interface selection	311
7.8.3	CAN subsystem configuration	311
7.8.4	LFASST and SIPI	311
7.8.5	DSPI configuration	313
7.8.6	FlexRay Configuration	316
7.8.7	PSI5 Controller Configuration	317
7.8.8	SENT receiver (SRX) configuration	317
7.8.9	LINFlexD — Configurations	320
7.9	Reset and boot modules	321
7.9.1	System Status and Configuration Module (SSCM) configuration	321
7.10	Safety modules	322
7.10.1	CRC – Number Of Contexts	322
7.10.2	MEMU configuration	323
7.10.3	Indirect Memory Access (IMA) configuration	331
7.10.4	FCCU configuration	335
7.10.5	Register protection (REG_PROT) configuration	336
7.11	Security modules	343
7.11.1	Password and Device Security Module (PASS) configuration	344
7.11.2	Tamper Detection Module (TDM) configuration	350
7.12	Production device vs. emulation device	356
8	Reset and Boot	359
8.1	Introduction	359
8.1.1	Boot Assist Flash	359

8.1.2	TEST flash memory block	359
8.1.3	UTEST flash memory block	360
8.1.4	Boot header	360
8.2	Modules used in reset sequence	360
8.2.1	Power Management Controller	360
8.2.2	Reset Generation Module	360
8.2.3	Mode Entry module	361
8.2.4	System Status and Configuration Module	361
8.2.5	Self-Test Control Unit	362
8.3	Reset sequence	362
8.3.1	Power-on and the Reset Generation Module	362
8.3.2	Power-up phase: power stabilization	364
8.3.3	PHASE0 Phase: analog supply initial configuration	366
8.3.4	PHASE1[DEST] Phase: temporization and monitoring setup	366
8.3.5	PHASE2[DEST] Phase: flash initial configuration	366
8.3.6	PHASE3[DEST] Phase: device configuration	367
8.3.7	IDLE[DEST] Phase: self test execution	368
8.3.8	PHASE1[FUNC] Phase: temporization and monitoring setup	369
8.3.9	PHASE2[FUNC] Phase: flash initial configuration	369
8.3.10	PHASE3[FUNC] Phase: device configuration monitoring	370
8.3.11	IDLE[FUNC] Phase	370
8.4	MC_RGM passes boot sequence control to the System Status and Control Module	370
8.4.1	Reset sequence flow based on initial device condition	371
8.4.2	Possible boot-up sequences	371
8.4.3	Sequence from MC_RGM IDLE to IOP, HSM, and boot CPU running code	372
8.4.4	Sequence from MC_RGM IDLE to IOP and Boot CPU running code (HSM disabled)	376
8.4.5	Path from MC_RGM IDLE to serial boot mode	379
8.4.6	Path from MC_RGM IDLE with IOP and HSM enabled to watchdog timer timeout	383
9	Device Configuration Format (DCF) Records	392
9.1	Introduction	392
9.2	DCF clients	392
9.3	DCF records	393

9.3.1	UTEST DCF records	396
9.4	DCF client table	397
10	Power management	419
10.1	Overview	419
10.1.1	Power management framework	421
10.1.2	Power management supply description	422
10.1.3	SPC57EM80xx power management controller overview	423
10.2	Low power mode support	424
10.2.1	Low power support and STOP mode implementation	424
10.3	Flash power requirements	426
10.4	Device trimming	426
10.5	Supply monitoring (POR and LVDs)	426
10.5.1	Power-on reset (POR)	426
10.5.2	Behavior of device LVD / HVD	426
10.5.3	Low Voltage Detection (LVD) and High Voltage Detection (HVD)	427
10.6	Power sequence	434
10.6.1	Power-up sequence	435
10.6.2	Power-down sequence	441
10.6.3	Brown-out management	441
10.6.4	Low voltage requirement during crank	442
11	Security	443
11.1	Basic security	443
11.2	Advanced security	443
11.3	Detailed security information	444
12	Calibration and Debug	445
12.1	Core debug support	445
12.2	Run control and memory access	445
12.2.1	Debug and Calibration Interface (DCI)	446
12.2.2	JTAG Data Communication (JDC)	449
12.2.3	Sequence Processing Unit (SPU)	449
12.3	Calibration interface	461
12.3.1	JTAG Master (JTAGM)	462
12.3.2	Debug LFAST	462

12.3.3	Development Trigger Semaphore (DTS)	462
12.4	Debug over CAN	463
12.5	Nexus Trace Aurora interface	463
12.5.1	Nexus Aurora overview	466
12.5.2	Nexus Aurora Router (NAR)	467
12.5.3	Nexus Aurora Link (NAL)	470
12.5.4	Nexus Aurora PHY (NAP)	470
12.6	Nexus clients	471
12.6.1	e200z4 Nexus 3	471
12.6.2	GTM Development Interface (GTMDI)	472
12.6.3	Nexus Crossbar Multi-master Client (NXMC)	472
12.7	Data trace filtering	474
13	Core Complex Overview	475
13.1	Overview of the e200z420n3, e200z419, and e200z425n3 cores	475
13.2	Features	476
13.3	Microarchitecture summary	477
13.3.1	Instruction unit features	479
13.3.2	Integer unit features	479
13.3.3	Load/Store unit features	479
13.3.4	EFPU2 Floating-Point Unit features	480
13.3.5	LSP Lightweight Signal Processing Unit features	480
13.3.6	MPU features	481
13.3.7	Cache features	481
13.3.8	Local memory features	481
13.3.9	Performance Monitor Unit Features	482
13.3.10	e200z420n3 and e200z425n3 system bus features	482
13.3.11	e200z419 system features	482
13.4	Availability of detailed documentation	483
14	Core e200z420n3 description	484
14.1	Overview of the e200z420n3 core	484
14.2	Register model	484
14.3	Dual-issue operation	487
14.4	Reservation instructions and cache interactions	488

14.5	Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)	488
14.6	Cache	491
14.6.1	Cache overview	491
14.6.2	L1 Cache Control and Status Register 0 (L1CSR0)	491
14.6.3	L1 Cache Control and Status Register 1 (L1CSR1)	494
14.6.4	L1 Cache Configuration Register 0 (L1CFG0)	496
14.6.5	L1 Cache Configuration Register 1 (L1CFG1)	497
14.6.6	Data Cache Software Coherency	498
14.6.7	Data Cache Hardware Coherency	498
14.6.8	Cache Invalidate by Set and Way	498
14.6.9	Cache EDC/ECC Parity Protection	500
14.6.10	Cache Error Injection	501
14.7	Exceptions	502
14.7.1	Exception Syndrome Register (ESR)	503
14.7.2	Machine State Register (MSR)	504
14.7.3	Machine Check Syndrome Register (MCSR)	505
14.7.4	Interrupt Vector Prefix Registers (IVPR)	508
14.7.5	Interrupt Definitions	509
14.8	Memory Protection Unit (MPU)	518
14.8.1	MPU Overview	518
14.8.2	Software Interface and MPU Instructions	518
14.8.3	MPU Read Entry Instruction (mpure)	518
14.8.4	MPU Write Entry Instruction (mpuwe)	519
14.8.5	MPU Synchronize Instruction (mpusync)	519
14.8.6	MMU/MPU Configuration Register (MMUCFG)	519
14.8.7	MPU0 Configuration Register (MPU0CFG)	520
14.8.8	MPU0 Control and Status Register 0 (MPU0CSR0)	521
14.8.9	MPU Assist Registers (MAS)	524
14.8.10	MAS Registers Summary	529
14.9	Local memories	529
14.9.1	Local Instruction and Data memory overview	529
14.9.2	Local memory control and configuration	529
14.10	End-to-End ECC support	543
14.10.1	End-to-End ECC control and configuration	543
15	Core e200z425n3 description	547

15.1	Overview of the e200z425n3 core	547
15.2	Register model	547
15.3	Dual-issue operation	550
15.4	Reservation instructions and cache interactions	551
15.5	Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)	551
15.6	Cache	554
15.6.1	Cache overview	554
15.6.2	L1 Cache Control and Status Register 0 (L1CSR0)	554
15.6.3	L1 Cache Control and Status Register 1 (L1CSR1)	555
15.6.4	L1 Cache Configuration Register 0 (L1CFG0)	557
15.6.5	L1 Cache Configuration Register 1 (L1CFG1)	557
15.6.6	Cache Invalidate by Set and Way	558
15.6.7	Cache EDC/ECC Parity Protection	559
15.6.8	Cache Error Injection	560
15.7	Exceptions	561
15.7.1	Exception Syndrome Register (ESR)	562
15.7.2	Machine State Register (MSR)	563
15.7.3	Machine Check Syndrome Register (MCSR)	564
15.7.4	Interrupt Vector Prefix Registers (IVPR)	567
15.7.5	Interrupt Definitions	568
15.8	Memory Protection Unit (MPU)	577
15.8.1	MPU Overview	577
15.8.2	Software Interface and MPU Instructions	577
15.8.3	MPU Read Entry Instruction (mpure)	577
15.8.4	MPU Write Entry Instruction (mpuwe)	578
15.8.5	MPU Synchronize Instruction (mpusync)	578
15.8.6	MMU/MPU Configuration Register (MMUCFG)	578
15.8.7	MPU0 Configuration Register (MPU0CFG)	579
15.8.8	MPU0 Control and Status Register 0 (MPU0CSR0)	580
15.8.9	MPU Assist Registers (MAS)	583
15.8.10	MAS Registers Summary	588
15.9	Local memories	588
15.9.1	Local Instruction and Data memory overview	588
15.9.2	Local memory control and configuration	588
15.10	End-to-End ECC support	602

	15.10.1	End-to-End ECC control and configuration	602
16		System Integration Unit Lite2 (SIUL2)	606
	16.1	Introduction	606
	16.1.1	Overview	606
	16.1.2	Features	608
	16.1.3	Register protection	608
	16.2	Memory map and register description	608
	16.2.1	Memory map	609
	16.2.2	Register descriptions	610
	16.3	Functional description	629
	16.3.1	General	629
	16.3.2	Pad control	629
	16.3.3	General purpose input or output pads (GPIO)	630
	16.3.4	External interrupts/DMA requests (EIRQ pins)	631
17		Crossbar Switch (XBAR)	635
	17.1	Introduction	635
	17.1.1	Features	635
	17.1.2	Memory map and register definition	635
	17.2	Functional description	641
	17.2.1	General operation	641
	17.2.2	Register coherency	642
	17.2.3	Arbitration	642
	17.3	Initialization/application information	643
18		Crossbar Integrity Checker (XBIC)	644
	18.1	Overview	644
	18.2	Features	644
	18.3	Block diagram	644
	18.4	External signal description	646
	18.5	Memory map and register definition	647
	18.5.1	Memory map	647
	18.5.2	Register descriptions	647
	18.6	Functional description	650

19	Crossbar Lock Control (CLC) module	653
19.1	Introduction	653
19.2	Block diagram	653
19.3	Memory Map and Register Definition	654
19.3.1	Crossbar Lock Control Global Enable (CLC_GEN) register	654
19.3.2	Crossbar Lock Control Master Enable (CLC_MEN) register	655
19.3.3	Crossbar Lock Control Slave Target (CLC_STGT) register	657
19.4	Functional description	660
19.5	Initialization/application information	661
20	Peripheral Bridge	662
20.1	Introduction	662
20.1.1	Features	662
20.1.2	General operation	662
20.2	Memory map/register definition	662
20.2.1	Master Privilege Registers (AIPS_MPRA & AIPS_MPRB)	664
20.2.2	Peripheral Access Control Registers (AIPS_PACRx)	665
20.2.3	Off-Platform Peripheral Access Control Registers (AIPS_OPACRx)	666
20.3	Functional description	669
20.3.1	Access support	669
21	System Memory Protection Unit (SMPU)	670
21.1	Overview	670
21.2	Block diagram	670
21.3	Features	671
21.4	Memory map and register definition	672
21.4.1	Memory map	672
21.4.2	Register descriptions	677
21.5	Functional description	685
21.5.1	Access evaluation macro	685
21.5.2	Putting it all together and error terminations	686
21.6	Initialization information	687
21.7	Application information	687
22	Intelligent AHB Gasket (IAHBG)	689
22.1	Introduction	689

22.2	Timing diagrams	689
22.2.1	1:1	689
22.2.2	2:1	691
22.2.3	1:2	695
22.2.4	64->32 Interface	699
22.2.5	32->64 Interface	699
23	Semaphores2 (SEMA42)	701
23.1	Introduction	701
23.1.1	Multi-core programming 101: software gates	701
23.1.2	Features	702
23.2	Memory map/register definition	704
23.2.1	Semaphores Gate n Register (SEMA4_GATE)	704
23.2.2	Semaphores (Secure) Reset Gate (SEMA4_RSTGT)	705
23.3	Functional description	706
24	Interrupt Controller (INTC)	709
24.1	Introduction	709
24.2	Block diagram	710
24.3	Features	711
24.4	Modes of operation	711
24.4.1	Software vector mode	711
24.4.2	Hardware vector mode	712
24.5	Memory map and register definition	712
24.5.1	Memory map	712
24.5.2	Register descriptions	713
24.6	Functional description	721
24.6.1	Interrupt request sources	721
24.6.2	Priority management	722
24.6.3	Handshaking with processor	724
24.7	Initialization/application information	726
24.7.1	Initialization flow	726
24.7.2	Interrupt exception handler	727
24.7.3	ISR, RTOS, and task hierarchy	729
24.7.4	Order of execution	729
24.7.5	Priority ceiling protocol	731

24.7.6	Selecting priorities according to request rates and deadlines	734
24.7.7	Software-settable interrupt requests	734
24.7.8	Lowering priority within an ISR	735
24.7.9	Negating an interrupt request outside of its ISR	735
24.7.10	Examining LIFO contents	736
24.8	Interrupt sources	736
25	Enhanced Direct Memory Access (eDMA)	737
25.1	Introduction	737
25.1.1	Features	737
25.2	Modes of operation	738
25.2.1	Normal mode	738
25.2.2	Debug mode	739
25.2.3	Wait mode	739
25.3	Memory map/register definition	739
25.3.1	Control Register (DMA_CR)	783
25.3.2	Error Status Register (DMA_ES)	785
25.3.3	Enable Request Register High (DMA_ERQH)	787
25.3.4	Enable Request Register Low (DMA_ERQL)	790
25.3.5	Enable Error Interrupt Register High (DMA_EEIH)	793
25.3.6	Enable Error Interrupt Register Low (DMA_EEIL)	796
25.3.7	Set Enable Request Register (DMA_SERQ)	799
25.3.8	Clear Enable Request Register (DMA_CERQ)	800
25.3.9	Set Enable Error Interrupt Register (DMA_SEEI)	801
25.3.10	Clear Enable Error Interrupt Register (DMA_CEEI)	801
25.3.11	Clear Interrupt Request Register (DMA_CINT)	802
25.3.12	Clear Error Register (DMA_CERR)	803
25.3.13	Set START Bit Register (DMA_SSRT)	803
25.3.14	Clear DONE Status Bit Register (DMA_CDNE)	804
25.3.15	Interrupt Request Register High (DMA_INTH)	805
25.3.16	Interrupt Request Register Low (DMA_INTL)	808
25.3.17	Error Register High (DMA_ERRH)	811
25.3.18	Error Register Low (DMA_ERRL)	814
25.3.19	Hardware Request Status Register High (DMA_HRSH)	817
25.3.20	Hardware Request Status Register Low (DMA_HRSL)	820
25.3.21	Channel <i>n</i> Priority Register (DMA_DCHPRIn)	823
25.3.22	Channel <i>n</i> Master ID Register (DMA_DCHMIDn)	824

25.3.23	TCD Source Address (DMA_TCDn_SADDR)	825
25.3.24	TCD Transfer Attributes (DMA_TCDn_ATTR)	826
25.3.25	TCD Signed Source Address Offset (DMA_TCDn_SOFF)	827
25.3.26	TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCDn_NBYTES_MLNO)	827
25.3.27	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCDn_NBYTES_MLOFFNO)	828
25.3.28	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCDn_NBYTES_MLOFFYES)	829
25.3.29	TCD Last Source Address Adjustment (DMA_TCDn_SLAST)	830
25.3.30	TCD Destination Address (DMA_TCDn_DADDR)	831
25.3.31	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_CITER_ELINKYES)	831
25.3.32	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_CITER_ELINKNO)	832
25.3.33	TCD Signed Destination Address Offset (DMA_TCDn_DOFF)	833
25.3.34	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCDn_DLASTSGA)	834
25.3.35	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_BITER_ELINKYES)	835
25.3.36	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_BITER_ELINKNO)	836
25.3.37	TCD Control and Status (DMA_TCDn_CSR)	836
25.4	Functional description	838
25.4.1	eDMA microarchitecture	839
25.4.2	eDMA basic data flow	840
25.4.3	Error reporting and handling	843
25.4.4	Channel preemption	845
25.4.5	eDMA performance	845
25.5	Initialization/application information	848
25.5.1	eDMA initialization	848
25.5.2	DMA programming errors	851
25.5.3	DMA Arbitration mode considerations	852
25.5.4	DMA transfer	853
25.5.5	eDMA TCDn status monitoring	856
25.5.6	Channel linking	858
25.5.7	Dynamic programming	859

26 DMACHMUX 862

26.1	Introduction	862
26.1.1	Overview	862
26.1.2	Features	862
26.1.3	Modes of Operation	863
26.2	External Signal Description	863
26.2.1	Overview	863
26.3	Memory Map and Register Definition	863
26.3.1	Register Descriptions	864
26.4	Functional description	865
26.4.1	DMA channels with periodic triggering capability	865
26.4.2	DMA Channels with no triggering capability	867
26.4.3	“Always Enabled” DMA Sources	867
26.5	Initialization/Application Information	868
26.5.1	Reset	868
26.5.2	Enabling and Configuring Sources	868
27	Clocking	873
27.1	Introduction	873
27.2	Clock generation	874
27.2.1	MC_CGM registers	876
27.3	System clock frequency limitations	878
27.3.1	JTAG frequencies	880
27.4	Default clock configuration	881
27.5	Clock sources	881
27.5.1	PLL	881
27.5.2	External oscillator (XOSC)	883
27.5.3	16 MHz internal RC oscillator (IRCOSC)	885
27.6	Peripheral clocks	886
27.6.1	HSM clock divider	888
27.6.2	PSI5 clock dividers	888
27.6.3	LFAST clocking	888
27.6.4	Ethernet clocking	889
27.6.5	FlexRay clocking	890
27.6.6	M_TTCAN/M_CAN clocking	891
27.6.7	Sigma-Delta ADC clocking	891
27.7	Clock monitoring	892

27.7.1	CMU configuration	892
27.7.2	PLL0 monitor	895
27.7.3	External oscillator (XOSC) monitor	895
27.7.4	Internal RC oscillator (IRCOSC) monitor	895
27.7.5	System clock monitors	895
27.8	Loss of system clock behavior	895
27.8.1	Loss of PLL/XOSC clock	895
27.8.2	Loss of IRCOSC clock	896
28	Dual PLL Digital Interface (PLLDIG)	897
28.1	Introduction	897
28.2	Block Diagram	897
28.3	Features	897
28.4	Modes of operation	897
28.4.1	Normal mode with reference, PLL0 or both PLLs enabled	898
28.5	Memory map and register definition	898
28.5.1	Memory map	898
28.5.2	Register descriptions	899
28.5.3	Register classification for safety requirements	909
28.6	Functional description	909
28.6.1	Input clock frequency	909
28.6.2	Clock configuration	909
28.6.3	Loss of lock	910
28.6.4	Frequency modulation	911
28.7	Initialization information	912
29	Clock Monitor Unit (CMU)	914
29.1	Introduction	914
29.1.1	Main features	914
29.2	Block diagram	915
29.3	Signals	915
29.4	Register description and memory map	916
29.4.1	Register descriptions	917
29.5	Functional description	921
29.5.1	Frequency meter	922
29.5.2	CLKMN0_RMT supervisor	922

	29.5.3 CLKMN1 supervisor	922
30	Clock Generation Module (MC_CGM)	923
30.1	Introduction	923
30.1.1	Overview	923
30.1.2	Features	925
30.2	External signal description	925
30.3	Memory map and register definition	925
30.3.1	Register descriptions	940
30.4	Functional description	989
30.4.1	System clock generation	989
30.4.2	Auxiliary clock generation	995
30.4.3	Dividers functional description	1004
31	OSC Digital Interface (XOSC)	1008
31.1	Introduction	1008
31.2	Functional description	1008
31.2.1	Oscillator power-down control and status	1008
31.2.2	Oscillator startup delay	1008
31.2.3	Oscillator clock available interrupt	1009
31.2.4	Oscillator bypass mode	1009
31.3	Memory map and register definition	1009
31.3.1	Register descriptions	1010
32	IRCOSC digital interface	1012
32.1	Functional description	1012
32.2	Memory map and register definition	1012
32.2.1	Register descriptions	1012
33	RAM controller (PRAM)	1016
33.1	Introduction	1016
33.2	SRAM controller memory map and register definition	1017
33.2.1	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	1017
33.3	Functional description	1018
33.3.1	Read / Write introduction	1018
33.3.2	Writes	1019

33.4	Initialization/application information	1020
33.5	Safety considerations	1021
33.5.1	Hsiao ECC algorithm	1021
33.5.2	Transaction monitor	1024
34	Flash memory controller	1026
34.1	Introduction	1026
34.2	Features	1026
34.3	Block diagrams	1027
34.4	Flash memory controller memory map	1028
34.4.1	Overview	1028
34.5	Functional description	1055
34.5.1	Basic interface protocol	1055
34.5.2	Access protections	1055
34.5.3	Read cycles - buffer miss	1056
34.5.4	Read cycles - buffer hit	1056
34.5.5	Error termination	1056
34.5.6	Flash error response operation	1056
34.5.7	Censorship	1057
34.5.8	Security module exclusive control	1057
34.5.9	Access pipelining	1058
34.5.10	Line read buffers and prefetch operation	1061
34.5.11	Instruction/Data prefetch triggering	1062
34.5.12	Per-Master prefetch triggering	1062
34.5.13	Buffer allocation	1062
34.5.14	PFlash calibration remap support	1063
34.5.15	AMU dedicated flash port	1069
34.5.16	Safety considerations	1069
34.5.17	e2eECC and EEPROM accesses	1073
34.5.18	Array integrity considerations	1073
35	Embedded Flash Memory (c55fmc)	1074
35.1	Introduction	1074
35.1.1	Overview	1074
35.1.2	Features	1075
35.1.3	Modes of operation	1075

35.2	Memory map and register definition	1075
35.2.1	Memory map	1076
35.2.2	Register Summary	1079
35.2.3	Register descriptions	1083
35.3	Functional Description	1131
35.3.1	User mode	1132
35.3.2	Low Power mode	1140
35.3.3	UTest mode	1140
35.4	Initialization information	1143
36	Decorated Storage Memory Controller (DSMC)	1145
36.1	Introduction	1145
36.2	DSMC Block Diagram	1147
36.3	Decorated Stores: st[b,h,w]d{cb}x rS,rB,rA	1149
36.3.1	Bit Field Insert (BFINS) into an 8, 16 or 32-bit Memory Container	1149
36.4	DSMC Timing Diagram	1154
36.5	DSMC Instantiations	1155
37	Flash Memory Programming and Configuration	1156
37.1	Selection of flash memory blocks for erase	1157
37.2	Non-secure write protection	1160
37.3	Secure write protection	1161
37.3.1	Implementing secure write protection	1163
37.3.2	Overriding secure write protection	1166
37.4	Secure read protection	1168
37.4.1	Implementing secure read protection	1169
37.4.2	Overriding secure read protection	1172
37.5	Debug port enable/disable	1173
37.6	Tamper detection	1173
37.6.1	Implementing tamper detection	1174
37.6.2	Creating the tamper detect diary	1174
37.6.3	Assigning blocks to Tamper Detection Regions (TDRs)	1175
37.6.4	Overriding tamper detection	1176
37.7	Implementing OTP	1177
37.8	Implementing test mode disable	1177
37.8.1	Unconditional test mode disable seal	1177

37.8.2	Passcode-protected test mode disable seal	1178
37.8.3	Selecting flash memory blocks for test mode disable seal	1178
37.9	Security configuration planning	1182
37.9.1	Hardware Security Module (HSM)	1182
37.9.2	Creating password groups	1183
37.9.3	Planning secure write protection	1183
37.9.4	Planning secure read protection	1183
37.9.5	Planning debug port enable/disable	1184
37.9.6	Planning OTP flash memory block assignment	1184
37.9.7	Planning factory test mode disable	1184
38	Analog-to-Digital Converters (ADC) Configuration	1185
38.1	ADC overview	1185
38.1.1	ADC subsystem block diagram	1185
38.1.2	Analog input pin multiplexing	1187
38.2	Configuration of ADC modules	1188
38.2.1	Sigma-Delta Analog-to-Digital Converter (SDADC)	1188
38.2.2	Successive Approximation Register Analog-to-Digital Converter (SARADC)	1201
39	Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface	1248
39.1	Introduction	1248
39.2	Overview	1248
39.3	Features	1250
39.4	Modes of operation	1250
39.4.1	Differential input mode	1250
39.4.2	Single-ended input mode	1250
39.4.3	External modulator mode	1250
39.5	External signal description	1251
39.5.1	Detailed signal descriptions	1251
39.6	Memory map and register descriptions	1252
39.6.1	Memory map	1252
39.6.2	Register descriptions	1252
39.7	Functional description	1266
39.7.1	Differential input mode	1266

	39.7.2	Single-ended input mode	1267
	39.7.3	External modulator mode	1267
	39.7.4	Low consumption mode	1267
	39.7.5	Analog input multiplexing and bias support	1267
	39.7.6	Programmable gain and decimation rate	1267
	39.7.7	High-pass filter support	1268
	39.7.8	Data conversion	1268
	39.7.9	Hardware triggering	1269
	39.7.10	Interrupt/DMA request support	1269
	39.7.11	Gain calibration support	1270
	39.7.12	Offset calibration support	1271
	39.8	Initialization information	1271
40		Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface	1273
	40.1	Introduction	1273
	40.2	Overview	1273
	40.3	Feature description	1275
	40.3.1	Main features	1275
	40.4	Functional description	1275
	40.4.1	Normal channel conversion	1275
	40.4.2	Injected channel conversion	1277
	40.4.3	Abort conversion	1278
	40.4.4	Analog conversion timings and reference selection	1279
	40.4.5	Test channel connection with internal analog channel	1280
	40.4.6	External channel mapping to internal analog channel	1281
	40.4.7	Programmable analog watchdog	1281
	40.4.8	DMA functionality	1282
	40.4.9	Interrupts	1283
	40.4.10	External decode signals selection and delay	1284
	40.4.11	Power down mode	1284
	40.5	Memory map and registers	1284
	40.5.1	Register descriptions	1286
	40.6	Start of conversion pulse delay	1333
41		Temperature Sensor	1336
	41.1	Introduction	1336

41.1.1	Signal Descriptions	1336
41.2	Functional description	1336
41.2.1	Temperature threshold detection (digital output generation)	1336
41.2.2	Linear temperature sensor (analog output generation)	1337
41.3	Temperature formula	1338
41.4	Calculating device temperature	1340
42	Timers Overview	1341
43	System Timer Module (STM)	1342
43.1	Introduction	1342
43.1.1	Overview	1342
43.1.2	Features	1342
43.1.3	Modes of operation	1342
43.2	External signal description	1342
43.3	Memory map and register definition	1342
43.3.1	Memory map	1342
43.3.2	Register descriptions	1343
43.4	Functional description	1346
44	Software Watchdog Timer (SWT)	1347
44.1	Introduction	1347
44.1.1	Overview	1347
44.1.2	Features	1347
44.1.3	Modes of operation	1347
44.2	External signal description	1347
44.3	Memory map and register definition	1347
44.3.1	Register descriptions	1348
44.4	Functional description	1353
45	Periodic Interrupt Timer (PIT)	1356
45.1	Introduction	1356
45.1.1	Block diagram	1356
45.1.2	Features	1357
45.2	Signal description	1357
45.2.1	Memory map/register description	1357

45.3	Functional description	1367
45.3.1	General	1367
45.3.2	Interrupts	1369
45.3.3	Chained timers	1369
45.4	Initialization and application information	1369
45.5	Example configuration for chained timers	1370
45.6	Example Configuration for the Lifetime Timer	1371
46	CAN Subsystem	1372
46.1	Introduction	1372
46.2	Features	1374
46.3	Modular CAN cores	1374
46.3.1	Features	1375
46.3.2	Block diagram	1376
46.3.3	Dual clock sources	1377
46.3.4	Dual interrupt lines	1377
46.3.5	Memory map and register descriptions	1377
46.3.6	Message RAM	1427
46.3.7	M_CAN functional description	1435
46.3.8	Timestamp generation	1444
46.3.9	Timeout counter	1444
46.3.10	Rx handling	1445
46.3.11	Dedicated Rx Buffers	1449
46.3.12	Debug on CAN Support	1449
46.3.13	Tx handling	1451
46.3.14	FIFO acknowledge handling	1454
46.3.15	Interface to DMA Controller	1455
46.4	Time Triggered Modular CAN (M_TTCAN) core	1455
46.4.1	Features	1456
46.4.2	Block Diagram	1457
46.4.3	Dual clock sources	1458
46.4.4	Dual interrupt lines	1458
46.4.5	Memory map and registers description	1458
46.4.6	Message RAM	1530
46.4.7	M_TTCAN functional description	1539
46.4.8	FIFO acknowledge handling	1555

46.4.9	M_TTCAN operations	1555
46.4.10	M_TTCAN configuration	1557
46.4.11	Message scheduling	1559
46.4.12	M_TTCAN gap control	1565
46.4.13	Stop watch	1566
46.4.14	Local time, cycle time, global time, and external clock synchronization	1566
46.4.15	M_TTCAN error level	1569
46.4.16	M_TTCAN message handling	1570
46.4.17	M_TTCAN interrupt and error handling	1573
46.4.18	Level 0	1573
46.4.19	Synchronization to external time schedule	1576
46.5	CAN RAM arbiter	1577
46.5.1	Features	1577
46.5.2	Functional overview using examples	1578
46.6	SRAM interface and memory organization	1578
46.6.1	Shared memory	1578
46.6.2	ECC controller	1579
46.6.3	Transmit message buffer locking mechanism	1580
46.7	Time Triggered CAN matrix and basic cycles management	1581
46.8	CAN nodes 1 and 2 I/Os sharing	1581
46.9	External signal description	1582
46.10	Memory map and register description	1582
46.10.1	Shared memory map	1582
47	Serial Interprocessor Interface (SIPI)	1585
47.1	Introduction	1585
47.1.1	Scalability	1585
47.2	Overview	1585
47.3	SIPI block diagram	1587
47.4	Feature description	1588
47.4.1	Main Features	1588
47.4.2	Standard features	1588
47.5	Functional description	1588
47.5.1	External signals	1588
47.5.2	Frame format	1588

47.6	Transfer types	1593
47.6.1	Read transfer	1593
47.6.2	Register read answer transfer	1594
47.6.3	Register write transfer	1595
47.6.4	Write Acknowledge transfer	1598
47.6.5	ID request response	1598
47.7	Transfer API and flow charts	1599
47.8	DMA programming sequence	1606
47.9	Modes of operation	1606
47.9.1	Initialization mode (INIT)	1606
47.9.2	Normal mode (NM)	1606
47.9.3	Module disable (MD)	1606
47.10	Errors	1607
47.10.1	Timeout error	1607
47.10.2	CRC error	1607
47.10.3	Maximum count reached error	1607
47.10.4	TID error	1607
47.10.5	Acknowledge error	1607
47.11	CRC calculation	1607
47.12	Interrupt logic	1608
47.13	SIPi control and status overview	1609
47.14	Memory map and register description	1610
47.14.1	Register descriptions	1612
48	Deserial Serial Peripheral Interface (DSPI)	1627
48.1	Introduction	1627
48.1.1	Block diagram	1627
48.1.2	Features	1628
48.1.3	DSPI configurations	1629
48.1.4	Modes of operation	1631
48.2	DSPI signal description	1631
48.2.1	PCS0/SS — Peripheral Chip Select/Slave Select	1632
48.2.2	PCS1 – PCS3 — Peripheral Chip Selects 1 – 3	1632
48.2.3	PCS4/MTRIG — Peripheral Chip Select 4/Master Trigger	1632
48.2.4	PCS5/PCSS — Peripheral Chip Select 5/Peripheral Chip Select Strobe	1632

48.2.5	PCS[6] – PCS[7] — Peripheral Chip Selects 6 – 7	1633
48.2.6	SIN — Serial Input	1633
48.2.7	SOUT — Serial Output	1633
48.2.8	SCK — Serial Clock	1633
48.2.9	HT — Hardware Trigger	1633
48.2.10	Memory map/register definition	1633
48.3	Register classification for safety	1691
48.4	Functional description	1692
48.4.1	Start and Stop of DSPI transfers	1693
48.4.2	Serial Peripheral Interface (SPI) configuration	1694
48.4.3	Deserial Serial Interface (DSI) configuration	1697
48.4.4	Combined Serial Interface (CSI) configuration	1704
48.4.5	DSPI baud rate and clock delay generation	1705
48.4.6	Transfer formats	1707
48.4.7	Continuous Serial Communications Clock	1717
48.4.8	Slave mode operation constraints	1719
48.4.9	Timed Serial Bus (TSB)	1719
48.4.10	Interleaved TSB (ITSB) Mode	1722
48.4.11	Parity generation and check	1725
48.4.12	Interrupts/DMA requests	1726
48.4.13	Power saving features	1729
48.5	Initialization/application information	1730
48.5.1	Managing DSPI queues	1730
48.5.2	Switching master and slave mode	1730
48.5.3	Initializing DSPI in Master/Slave Modes	1730
48.5.4	Baud rate settings	1731
48.5.5	Delay settings	1731
48.5.6	Calculation of FIFO pointer addresses	1732
49	LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications	1735
49.1	Introduction	1735
49.2	Block diagram	1735
49.3	External signals	1735
49.3.1	LFAST operating data rates	1736
49.4	LFAST frame structure	1736

49.5	Features	1739
49.6	Memory map and register definition	1739
49.6.1	Memory map	1739
49.6.2	Register descriptions	1742
49.6.3	Register safety classification requirements	1773
49.7	Functional description	1774
49.7.1	Startup procedure	1774
49.7.2	Line Receiver (LR)	1776
49.7.3	Transmit Controller	1785
49.7.4	CTS mode support	1789
49.7.5	Frames supported	1790
49.7.6	Frame flow	1790
49.7.7	Test and Debug Support	1796
49.7.8	Interrupts	1803
49.8	Packet memory	1805
49.9	Resets	1806
49.10	Clocks	1806
49.10.1	Clocking strategy	1806
49.10.2	Slow speed clock	1807
49.10.3	Rx controller clocks	1810
49.10.4	Tx controller clocks	1811
50	Fast Ethernet Controller (FEC)	1813
50.1	Introduction	1813
50.1.1	Block diagram	1813
50.1.2	Features	1816
50.2	Modes of operation	1816
50.2.1	Full and half duplex operation	1816
50.2.2	Interface options	1817
50.2.3	Address recognition options	1817
50.2.4	Internal loopback	1817
50.3	External signal description	1818
50.4	Memory map and register definition	1819
50.4.1	Interrupt Event Register (EIR)	1822
50.4.2	Interrupt Mask Register (EIMR)	1824
50.4.3	Receive Descriptor Active Register (RDAR)	1825

50.4.4	Transmit Descriptor Active Register (TDAR)	1826
50.4.5	Ethernet Control Register (ECR)	1827
50.4.6	MII Management Frame Register (MMFR)	1827
50.4.7	MII Speed Control Register (MSCR)	1829
50.4.8	MIB Control Register (MIBC)	1830
50.4.9	Receive Control Register (RCR)	1831
50.4.10	Transmit Control Register (TCR)	1832
50.4.11	Physical Address Lower Register (PALR)	1833
50.4.12	Physical Address Upper Register (PAUR)	1834
50.4.13	Opcode/Pause Duration Register (OPD)	1834
50.4.14	Descriptor Individual Upper Address Register (IAUR)	1835
50.4.15	Descriptor Individual Lower Address Register (IALR)	1835
50.4.16	Descriptor Group Upper Address Register (GAUR)	1836
50.4.17	Descriptor Group Lower Address Register (GALR)	1836
50.4.18	Transmit FIFO Watermark Register (TFWR)	1837
50.4.19	FIFO Receive Bound Register (FRBR)	1837
50.4.20	FIFO Receive Start Register (FRSR)	1838
50.4.21	Receive Descriptor Ring Start Register (ERDSR)	1838
50.4.22	Transmit Buffer Descriptor Ring Start Register (ETSDR)	1839
50.4.23	Receive Buffer Size Register (EMRBR)	1839
50.4.24	Count of frames not counted correctly (RMON_T_DROP)	1840
50.4.25	RMON Tx packet count (RMON_T_PACKETS)	1840
50.4.26	RMON Tx broadcast packets (RMON_T_BC_PKT)	1841
50.4.27	RMON Tx multicast packets (RMON_T_MC_PKT)	1841
50.4.28	RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN) ..	1842
50.4.29	RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)	1842
50.4.30	RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)	1842
50.4.31	RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)	1843
50.4.32	RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB) ...	1843
50.4.33	RMON Tx collision count (RMON_T_COL)	1843
50.4.34	RMON Tx 64 byte packets (RMON_T_P64)	1844
50.4.35	RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)	1844
50.4.36	RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)	1844
50.4.37	RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)	1845
50.4.38	RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)	1845
50.4.39	RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047) ..	1845

50.4.40	RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)	1846
50.4.41	RMON Tx Octets (RMON_T_OCTETS)	1846
50.4.42	Count of transmitted frames not counted correctly (IEEE_T_DROP)	1846
50.4.43	Frames transmitted OK (IEEE_T_FRAME_OK)	1847
50.4.44	Frames transmitted with single collision (IEEE_T_1COL)	1847
50.4.45	Frames transmitted with multiple collisions (IEEE_T_MCOL)	1847
50.4.46	Frames transmitted after deferral delay (IEEE_T_DEF)	1848
50.4.47	Frames transmitted with late collision (IEEE_T_LCOL)	1848
50.4.48	Frames transmitted with excessive collisions (IEEE_T_EXCOL) . . .	1848
50.4.49	Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR) . . .	1849
50.4.50	Frames transmitted with carrier sense error (IEEE_T_CSERR) . . .	1849
50.4.51	Frames transmitted with SQE error (IEEE_T_SQE)	1849
50.4.52	Flow control pause frames transmitted (IEEE_T_FDXFC)	1850
50.4.53	Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)	1850
50.4.54	Count of received frames not counted correctly (RMON_R_DROP) .	1850
50.4.55	RMON Rx packet count (RMON_R_PACKETS)	1851
50.4.56	RMON Rx broadcast packets (RMON_R_BC_PKT)	1851
50.4.57	RMON Rx multicast packets (RMON_R_MC_PKT)	1852
50.4.58	RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN) .	1852
50.4.59	RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)	1852
50.4.60	RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)	1853
50.4.61	RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)	1853
50.4.62	RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB) . . .	1853
50.4.63	Reserved (RMON_R_RESVD_0)	1854
50.4.64	RMON Rx 64 byte packets (RMON_R_P64)	1854
50.4.65	RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)	1854
50.4.66	RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)	1855
50.4.67	RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)	1855
50.4.68	RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023) . . .	1855
50.4.69	RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047) . .	1856
50.4.70	RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048) . . .	1856
50.4.71	RMON Rx octets (RMON_R_OCTETS)	1856
50.4.72	Count of received frames not counted correctly (IEEE_R_DROP) . .	1857
50.4.73	Frames received OK (IEEE_R_FRAME_OK)	1857
50.4.74	Frames received with CRC error (IEEE_R_CRC)	1857

50.4.75	Frames received with alignment error (IEEE_R_ALIGN)	1858
50.4.76	Receive FIFO overflow count (IEEE_R_MACERR)	1858
50.4.77	Flow control pause frames received (IEEE_R_FDXFC)	1858
50.4.78	Octet count for frames received without error (IEEE_R_OCTETS_OK)	1859
50.5	Functional description	1859
50.5.1	MII data frame	1859
50.5.2	MII management frame structure	1860
50.5.3	Buffer descriptors	1861
50.5.4	Initialization sequence	1866
50.5.5	User initialization (Prior to Setting ECR[ETHER_EN])	1866
50.5.6	Microcontroller initialization	1867
50.5.7	User initialization (after setting ECR[ETHER_EN])	1867
50.5.8	Network interface options	1867
50.5.9	FEC frame transmission	1869
50.5.10	FEC frame reception	1870
50.5.11	Ethernet address recognition	1871
50.5.12	Hash algorithm	1873
50.5.13	Full duplex flow control	1876
50.5.14	Inter-Packet Gap (IPG) time	1876
50.5.15	Collision managing	1877
50.5.16	MII internal and external loopback	1877
50.5.17	RMII loopback	1877
50.5.18	RMII echo	1878
50.5.19	Ethernet error-managing procedure	1878
51	FlexRay Communication Controller (FlexRay)	1880
51.1	Introduction	1880
51.1.1	Reference	1880
51.1.2	Glossary	1880
51.1.3	Color coding	1881
51.1.4	Overview	1881
51.1.5	Features	1883
51.1.6	Modes of operation	1884
51.2	External Signal Description	1886
51.2.1	Detailed signal description	1887
51.3	Controller Host Interface clocking	1887

51.4	Protocol engine clocking	1888
51.4.1	Oscillator clocking	1888
51.4.2	PLL clocking	1888
51.5	Memory map and register description	1888
51.5.1	Memory map	1888
51.5.2	Register descriptions	1895
51.6	Functional description	1981
51.6.1	Message buffer concept	1981
51.6.2	Physical message buffer	1981
51.6.3	Message Buffer Types	1982
51.6.4	FlexRay memory area layout	1988
51.6.5	Physical message buffer description	1991
51.6.6	Individual message buffer functional description	2001
51.6.7	Individual message buffer search	2017
51.6.8	Individual message buffer reconfiguration	2020
51.6.9	Receive FIFOs	2021
51.6.10	Channel device modes	2027
51.6.11	External clock synchronization	2029
51.6.12	Sync frame ID and sync frame deviation tables	2030
51.6.13	MTS generation	2033
51.6.14	Key slot transmission	2034
51.6.15	Sync frame filtering	2034
51.6.16	Strobe signal support	2035
51.6.17	Timer support	2036
51.6.18	Slot status monitoring	2037
51.6.19	System bus access	2041
51.6.20	Interrupt support	2042
51.6.21	Lower bit rate support	2046
51.6.22	PE data memory (PE DRAM)	2047
51.6.23	CHI lookup-table memory (CHI LRAM)	2048
51.6.24	Memory content error detection	2049
51.6.25	Memory error injection	2053
51.6.26	StopWatch function	2055
51.7	Application information	2057
51.7.1	Module configuration	2057
51.7.2	Initialization sequence	2058
51.7.3	Memory error injection out of POC:default config	2059

51.7.4	Shutdown sequence	2059
51.7.5	Number of usable message buffers	2060
51.7.6	Protocol Control Command Execution	2061
51.7.7	Message buffer search on simple message buffer configuration	2062
52	Inter-Integrated Circuit (I²C)	2065
52.1	Overview	2065
52.2	Introduction to I ² C	2065
52.2.1	Definition: I ² C module	2065
52.2.2	Advantages of the I ² C bus	2065
52.2.3	Module block diagram	2065
52.2.4	Features	2067
52.2.5	Modes of operation	2067
52.2.6	Definition: I ² C conditions	2068
52.3	External signal descriptions	2069
52.3.1	Signal overview	2069
52.3.2	Detailed external signal descriptions	2069
52.4	Memory map and register definition	2069
52.4.1	I ² C memory map	2069
52.4.2	Register accessibility	2070
52.4.3	Register figure conventions	2070
52.4.4	I ² C Bus Address Register (IBAD)	2071
52.4.5	I ² C Bus Frequency Divider Register (IBFD)	2071
52.4.6	I ² C Bus Control Register (IBCR)	2072
52.4.7	I ² C Bus Status Register (IBSR)	2073
52.4.8	I ² C Bus Data I/O Register (IBDR)	2075
52.4.9	I ² C Bus Interrupt Config Register (IBIC)	2076
52.4.10	I ² C Bus Debug Register (IBDBG)	2076
52.5	Functional description	2077
52.5.1	Notes about module operation	2077
52.5.2	Transactions	2077
52.5.3	Arbitration procedure	2080
52.5.4	Clock behavior	2081
52.5.5	Interrupts	2090
52.5.6	IPG DEBUG mode	2091
52.5.7	DMA interface	2092

52.6	Initialization/application information	2093
52.6.1	Recommended interrupt service flow	2093
52.6.2	General programming guidelines (for both master and slave mode)	2095
52.6.3	Programming guidelines specific to master mode	2096
52.6.4	Programming guidelines specific to slave mode	2098
52.6.5	DMA application information	2098
53	Peripheral Sensor Interface (PSI5)	2104
53.1	Introduction	2104
53.1.1	Overview	2104
53.1.2	Features	2106
53.1.3	Modes of operation	2107
53.2	External signal description	2112
53.3	Memory map and register description	2112
53.3.1	Memory map	2112
53.3.2	Register description	2135
53.4	Functional description	2190
53.4.1	Sensor-to-ECU communication	2190
53.4.2	Data Link Layer Data bit decoding	2191
53.4.3	ECU-to-sensor communication	2212
53.4.4	Tx/Rx scenario	2222
53.4.5	Interrupt handling	2223
53.4.6	DMA support	2225
53.4.7	Message read through interrupt generation	2226
53.5	Clock and Reset	2227
53.5.1	Latency and Jitter related requirements	2229
53.5.2	Reset	2229
53.6	Initialization information	2229
53.6.1	Initialization	2229
53.7	Design Assumptions and Limitations	2230
53.7.1	Assumptions	2230
53.7.2	Limitations	2230
53.8	Miscellaneous descriptions	2231
53.8.1	Internal SYNC Pulse generation and coordination across PSI5 channels	2231
53.8.2	Valid states for integrated sync pulse generator	2234

	53.8.3	Implementation of EI Diagnostic Register	2236
54		Peripheral Sensor Interface-Support Module (PSI5-S)	2237
	54.1	Introduction	2237
	54.1.1	Overview	2237
	54.1.2	Features	2239
	54.1.3	PSI5-S Global Modes of operation	2240
	54.1.4	PSI5-S System Modes of operation	2243
	54.2	External signal description	2244
	54.3	Memory map and register description	2245
	54.3.1	Memory map	2245
	54.3.2	UART register configuration	2255
	54.3.3	PSI5-S Register description	2274
	54.4	Functional description	2318
	54.4.1	PSI5-S UART Message Structure	2319
	54.4.2	PSI5-S Received Message handling	2321
	54.4.3	ECU-to-sensor communication	2328
	54.4.4	TimeStamp	2335
	54.4.5	PSI5-S Watchdog Operation	2336
	54.4.6	Interrupt handling	2338
	54.5	PSI5-S Clock and Reset	2341
55		SENT Receiver (SRX)	2343
	55.1	Introduction	2343
	55.1.1	Features	2343
	55.1.2	Modes of operation	2344
	55.1.3	Block diagram	2345
	55.1.4	Design overview	2346
	55.2	External signal description	2347
	55.2.1	SENT_RX [(CH-1):0]	2347
	55.3	Memory map and register definition	2347
	55.3.1	Memory map	2347
	55.3.2	Register descriptions	2349
	55.3.3	Register Classification for Safety	2383
	55.4	Functional description	2384
	55.4.1	Initialization sequence	2384

55.4.2	DMA read logic	2384
55.4.3	Message reading via interrupts	2389
55.4.4	Overflow behavior	2390
55.4.5	Adjustment for variation in sensor (Tx) clock	2391
55.4.6	Input programmable filter	2391
55.4.7	Receiver diagnostics	2392
55.4.8	Time stamp logic	2394
55.4.9	Bus Idle Diagnostic	2395
55.5	Clocks and resets	2395
55.5.1	Clocking strategy	2395
55.5.2	System bus clock requirements	2396
55.5.3	High frequency receiver clock (protocol clock) requirements	2397
56	LINFlexD	2398
56.1	Introduction	2398
56.2	Main features	2400
56.2.1	LIN mode features	2400
56.2.2	UART mode features	2401
56.2.3	Standard features	2401
56.3	Functional description	2401
56.3.1	LIN protocol	2401
56.3.2	LINFlexD features	2403
56.3.3	Timer	2419
56.3.4	UART mode	2420
56.3.5	DMA interface	2425
56.4	Memory map and register description	2443
56.4.1	Memory map	2443
56.4.2	Register description	2445
56.4.3	Register map	2484
56.5	Programming considerations	2487
56.5.1	Master node	2487
56.5.2	Slave node	2488
56.5.3	Extended frames	2493
56.5.4	Timeout	2493
56.5.5	UART mode	2494
56.5.6	Interrupts	2494

56.6	External signal description	2496
56.6.1	Interface signals summary	2496
57	Reset Generation Module (MC_RGM)	2500
57.1	Introduction	2500
57.1.1	Overview	2500
57.1.2	Features	2502
57.1.3	Reset sources	2502
57.2	External signal description	2503
57.3	Memory map and register definition	2503
57.3.1	Register descriptions	2514
57.4	Functional description	2542
57.4.1	Reset state machine	2542
57.4.2	'Destructive' resets	2545
57.4.3	External reset	2545
57.4.4	'Functional' resets	2546
57.4.5	Alternate event generation	2547
57.4.6	'Functional' reset escalation	2547
57.4.7	'Destructive' reset escalation	2548
57.4.8	Individual peripheral resets	2548
58	Boot Assist Flash (BAF)	2549
58.1	Introduction	2549
58.2	Memory map	2549
58.2.1	BAF image header	2549
58.3	Functional description	2551
58.3.1	Boot modes	2551
58.3.2	Device initialization	2551
58.3.3	Flow of control	2551
58.3.4	Optionally remap calibration RAM	2552
58.3.5	Optionally wait for HSM	2553
58.3.6	Initialization of BAF DCF clients	2554
58.3.7	Search boot header and boot options	2560
58.3.8	Production disable activation protocol and implementation	2561
58.3.9	Optionally perform a serial boot	2564
58.3.10	UART serial boot configuration	2568

58.4	Resources	2571
59	System Status and Configuration Module (SSCM)	2572
59.1	Introduction	2572
59.1.1	Overview	2572
59.1.2	Glossary	2572
59.1.3	Features	2573
59.1.4	Modes of operation	2573
59.2	External signal description	2573
59.3	Memory map and register definition	2573
59.3.1	Register descriptions	2574
59.4	Functional description	2579
59.4.1	DCF mechanism	2579
59.4.2	ECC error monitoring	2581
59.4.3	Boot mode functionality	2581
59.4.4	BAF configuration	2581
59.4.5	HSM boot header search	2582
59.4.6	Life Cycle	2582
59.5	Initialization and application information	2584
59.5.1	Reset	2584
59.6	Additional safety measures	2584
59.6.1	Spurious reset protection	2584
60	Power Management Controller digital interface (PMC_dig)	2585
60.1	Introduction	2585
60.2	Block diagram	2585
60.3	Memory Map/Register Definition	2587
60.3.1	Register descriptions	2589
60.4	Analog PMC interface	2649
60.5	Temperature Sensor Interface Logic	2649
60.6	Standby RAM Regulator Interface Logic	2650
60.7	Power On Reset (MC_RGM Phase Gates)	2650
60.8	Flash interface (DCF client usage)	2650
60.9	Other module interfaces	2651
61	Power Control Unit (MC_PCU)	2653

61.1	Introduction	2653
61.2	Memory map and register definition	2654
61.2.1	Memory map	2654
61.2.2	Register descriptions	2654
61.3	Functional description	2655
62	Mode Entry Module (MC_ME)	2656
62.1	Introduction	2656
62.1.1	Overview	2656
62.1.2	Features	2658
62.1.3	Modes of operation	2658
62.2	External signal description	2659
62.3	Memory map and register definition	2659
62.3.1	Memory map	2660
62.3.2	Register description	2672
62.4	Functional description	2707
62.4.1	Mode transition request	2707
62.4.2	Modes details	2708
62.4.3	Mode transition process	2712
62.4.4	Protection of Mode Configuration Registers	2720
62.4.5	Mode Transition Interrupts	2720
62.4.6	Peripheral Clock Gating	2722
62.4.7	Application Example	2723
63	e200z420n3 Core Debug Support	2725
63.1	Overview	2725
63.1.1	Software debug facilities	2725
63.1.2	Additional debug facilities	2726
63.1.3	Hardware debug facilities	2726
63.1.4	Sharing debug resources by software/hardware	2727
63.2	Software debug events and exceptions	2728
63.2.1	Instruction Address Compare event	2729
63.2.2	Data Address Compare event	2730
63.2.3	Linked Instruction Address Compare and Data Address Compare events	2739
63.2.4	Trap debug event	2739

63.2.5	Branch Taken debug event	2739
63.2.6	Instruction Complete debug event	2739
63.2.7	Interrupt Taken debug event	2740
63.2.8	Critical Interrupt Taken debug event	2740
63.2.9	Return debug event	2740
63.2.10	Critical Return Debug Event	2741
63.2.11	External debug event	2741
63.2.12	Unconditional debug event	2741
63.2.13	Performance Monitor Interrupt debug event	2741
63.3	Debug registers	2741
63.3.1	Debug Address and Value registers	2742
63.3.2	Debug Control and Status registers	2743
63.3.3	External Debug Resource Allocation Control (EDBRAC0) register	2763
63.3.4	Debug Event Select register (DEVENT) register	2769
63.3.5	Debug Data Acquisition Message (DDAM) register	2770
63.4	Using debug resources for stack limit checking	2770
63.5	External debug support	2772
63.5.1	External debug registers	2773
63.5.2	OnCE introduction	2778
63.5.3	JTAG/OnCE pins	2781
63.5.4	OnCE internal interface signals	2781
63.5.5	OnCE interface signals	2782
63.5.6	OnCE controller and serial interface	2784
63.5.7	Access to debug resources	2791
63.5.8	Methods of entering Debug mode	2793
63.5.9	CPU Status and Control Scan Chain Register (CPUSCR)	2794
63.5.10	Reserved registers (Reserved)	2800
63.6	Watchpoint support	2800
63.7	MPU operation during debug	2803
63.8	Cache array access during debug	2803
63.9	Basic steps for enabling, using, and exiting External Debug mode	2804
64	e200z425n3 Core Debug Support	2806
64.1	Overview	2806
64.1.1	Software debug facilities	2806
64.1.2	Additional debug facilities	2807

64.1.3	Hardware debug facilities	2807
64.1.4	Sharing debug resources by software/hardware	2808
64.2	Software debug events and exceptions	2809
64.2.1	Instruction Address Compare event	2810
64.2.2	Data Address Compare event	2811
64.2.3	Linked Instruction Address Compare and Data Address Compare events	2820
64.2.4	Trap debug event	2820
64.2.5	Branch Taken debug event	2820
64.2.6	Instruction Complete debug event	2820
64.2.7	Interrupt Taken debug event	2821
64.2.8	Critical Interrupt Taken debug event	2821
64.2.9	Return debug event	2821
64.2.10	Critical Return Debug Event	2822
64.2.11	External debug event	2822
64.2.12	Unconditional debug event	2822
64.2.13	Performance Monitor Interrupt debug event	2822
64.3	Debug registers	2822
64.3.1	Debug Address and Value registers	2823
64.3.2	Debug Control and Status registers	2824
64.3.3	External Debug Resource Allocation Control (EDBRAC0) register	2844
64.3.4	Debug Event Select register (DEVENT) register	2851
64.3.5	Debug Data Acquisition Message (DDAM) register	2851
64.4	Using debug resources for stack limit checking	2852
64.5	External debug support	2854
64.5.1	External debug registers	2855
64.5.2	OnCE introduction	2860
64.5.3	JTAG/OnCE pins	2863
64.5.4	OnCE internal interface signals	2863
64.5.5	OnCE interface signals	2864
64.5.6	OnCE controller and serial interface	2866
64.5.7	Access to debug resources	2873
64.5.8	Methods of entering Debug mode	2875
64.5.9	CPU Status and Control Scan Chain Register (CPUSCR)	2876
64.5.10	Reserved registers (Reserved)	2882
64.6	Watchpoint support	2882

64.7	MPU operation during debug	2885
64.8	Cache array access during debug	2886
64.9	Basic steps for enabling, using, and exiting External Debug mode . . .	2886
65	Debug and Calibration Interface (DCI)	2888
65.1	Introduction	2888
65.1.1	Features	2888
65.1.2	Overview	2888
65.1.3	Operating modes	2890
65.2	External signal description	2902
65.3	Register description	2903
65.3.1	DCI control register (DCI_CR)	2903
65.3.2	DCI EVTx pin multiplexing control register (DCI_PINCR)	2904
65.4	Functional description	2905
65.4.1	DCI mode transition	2905
65.4.2	LFAST LVDS pad fault	2906
66	JTAG Controller (JTAGC)	2908
66.1	Introduction	2908
66.1.1	Block Diagram	2908
66.1.2	Features	2909
66.1.3	Modes of operation	2909
66.2	External signal description	2910
66.2.1	TCK—Test clock input	2910
66.2.2	TDI—Test data input	2910
66.2.3	TDO—Test data output	2910
66.2.4	TMS—Test mode select	2910
66.2.5	JCOMP—JTAG compliancy	2910
66.3	Register description	2911
66.3.1	Instruction register	2911
66.3.2	Bypass register	2911
66.3.3	Device identification register	2911
66.3.4	JTAG_PASSWORD register	2912
66.3.5	Boundary scan register	2913
66.4	Functional description	2913
66.4.1	JTAGC reset configuration	2913

	66.4.2	IEEE 1149.1-2001 (JTAG) Test Access Port	2913
	66.4.3	TAP controller state machine	2913
	66.4.4	JTAGC block instructions	2915
	66.4.5	Boundary scan	2918
	66.5	Initialization/Application information	2918
67		IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)	2920
	67.1	References	2920
	67.2	Abbreviations	2920
	67.3	Introduction	2921
	67.3.1	Types of operation	2922
	67.3.2	Deployment by class	2922
	67.3.3	1149.7 TAP signals	2922
	67.3.4	TAP.7 architecture	2923
	67.3.5	Protocols	2924
	67.3.6	Operating models	2925
	67.4	CJTAG implementation summary	2926
	67.4.1	T0 Functions	2926
	67.4.2	T1 functions	2926
	67.4.3	T2 functions	2926
	67.4.4	T3 functions	2926
	67.4.5	T4 functions	2927
	67.5	Ancillary services	2927
	67.5.1	Overview	2927
	67.5.2	Resets	2927
	67.5.3	Start-up Options	2930
	67.5.4	RSU operation	2931
	67.5.5	TAPC State Machine	2932
	67.6	EPU (Extended Protocol Unit) Operation	2933
	67.6.1	EPU Operation	2933
	67.6.2	EPU Registers	2935
	67.6.3	EPU Commands	2945
	67.6.4	EPU operating states	2952
	67.6.5	System and EPU Paths	2953
	67.7	APU (Advanced Protocol Unit)	2953
	67.7.1	Overview	2953

	67.7.2	Operation	2956
	67.7.3	Escape sequences	2957
	67.7.4	Signal behaviors	2957
	67.7.5	APU Functions	2957
	67.7.6	Configuration Change Packets (CP)	2964
	67.7.7	Scan Packet	2964
	67.7.8	SP Format	2964
68		JTAG Data Communication (JDC)	2970
	68.1	Introduction	2970
	68.2	Overview	2970
	68.3	Register definition	2971
	68.3.1	Register descriptions	2971
	68.3.2	Non-Memory mapped register definition	2973
	68.4	Functional description	2974
	68.4.1	IPS register access	2975
	68.4.2	JTAG register access	2975
	68.5	Use case example	2975
69		Sequence Processing Unit (SPU)	2977
	69.1	Introduction	2977
	69.1.1	SPU block diagram	2977
	69.1.2	Overview	2978
	69.1.3	Features	2979
	69.2	SPU actions	2981
	69.3	Operation mode	2982
	69.3.1	SPU enable/disable	2982
	69.4	SPU module interface	2982
	69.5	Register definition	2982
	69.5.1	Register descriptions	2987
	69.6	Functional description	3022
	69.6.1	Input Mux unit	3022
	69.6.2	Processor exception vector encoding	3023
	69.6.3	State logic unit (SLU)	3024
	69.6.4	Sequence formation	3025
	69.6.5	Performance counters/timers unit	3028

	69.6.6	Action unit	3030
	69.6.7	Optional trace group	3030
70		JTAG Master (JTAGM)	3031
	70.1	Introduction	3031
	70.1.1	Overview	3031
	70.1.2	Feature description	3032
	70.2	Functional description	3032
	70.2.1	Module architecture	3032
	70.2.2	JTAGM JCOMP usage in LFAST	3033
	70.2.3	JTAGM data error detection	3034
	70.2.4	JTAGM message tagging	3034
	70.2.5	JTAGM Ready signal	3034
	70.2.6	EVTO and EVTI signals	3035
	70.2.7	JTAGM configuration and status monitoring	3035
	70.2.8	JTAGM to DCI serial interface	3036
	70.2.9	JTAGM data handling and CRC calculation in LFAST mode	3036
	70.2.10	Software interface	3037
	70.3	Modes of operation	3038
	70.4	Memory map and register definition	3038
	70.4.1	Register descriptions	3038
71		LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed Debug	3047
	71.1	Introduction	3047
	71.2	Block diagram	3047
	71.3	External signals	3048
	71.3.1	LFAST operating data rates	3048
	71.4	LFAST frame structure	3048
	71.5	Features	3051
	71.6	Memory map and register definition	3051
	71.6.1	Memory map	3051
	71.6.2	Register descriptions	3054
	71.6.3	Register safety classification requirements	3076
	71.7	Functional description	3076
	71.7.1	Startup procedure	3076

71.7.2	LFAST Interface enable signal (lfast_sysclk_en)	3077
71.7.3	Line Receiver (LR)	3077
71.7.4	Transmit Controller	3086
71.7.5	Frames supported	3090
71.7.6	Frame flow	3090
71.7.7	Test and Debug Support	3093
71.8	Packet memory	3100
71.9	Resets	3100
71.10	Clocks	3102
71.10.1	Clocking strategy	3102
71.10.2	Slow speed clock	3102
71.10.3	Rx controller clocks	3105
71.10.4	Tx controller clocks	3106
72	Development Trigger Semaphore (DTS)	3108
72.1	Introduction	3108
72.2	Overview	3108
72.3	DTS device connections	3110
72.3.1	DTS register access	3111
72.4	Memory map and register definition	3111
72.4.1	Register descriptions	3112
72.5	Example application	3115
73	Nexus Aurora Router (NAR)	3117
73.1	Introduction	3117
73.2	Overview	3118
73.3	Features	3118
73.4	Register definition	3118
73.4.1	Register descriptions	3119
73.4.2	Message filtering registers	3122
73.4.3	Trace memory bus configuration registers	3125
73.4.4	Suppress mode triggers register (NAR_STCR)	3128
73.4.5	Receive queue client disable register (NAR_CDR)	3130
73.4.6	Trace Memory fill level and partition configuration register (NAR_AHFPAR)	3131
73.5	Functional description	3132

73.5.1	Reset/Startup	3132
73.5.2	Operation modes	3133
73.5.3	Data reception and arbitration	3135
73.5.4	Suppress mode	3136
73.5.5	Stall detection	3137
73.5.6	Output arbitration	3138
73.5.7	Message translation and formatting	3139
73.5.8	Aurora interface	3140
73.5.9	Trace memory bus output port usage	3140
73.5.10	Internal message generation	3141
73.5.11	Timestamp function	3145
73.5.12	Overlay and internal trace memory full status generation	3146
73.5.13	Configuration/debug interface JTAG (1149.1)	3147
73.5.14	Trace memory bus controller	3150
73.5.15	Legacy client interface controller	3152
73.6	Initialization information	3153
73.7	Application information	3153
74	e200z420n3 Nexus 3 Module	3154
74.1	Introduction	3154
74.1.1	General Description	3154
74.1.2	Terms and Definitions	3154
74.1.3	Feature List	3155
74.1.4	Functional Block Diagram	3157
74.2	Enabling Nexus 3 Operation	3157
74.2.1	Interaction with Low Power Modes	3158
74.3	TCODEs supported	3158
74.4	Nexus 3 Programmer's Model	3164
74.4.1	Client Select Control (CSC) register	3165
74.4.2	Port Configuration Register (PCR) — reference only	3166
74.4.3	Nexus Development Control Register 1 (DC1)	3167
74.4.4	Nexus Development Control Registers 2 & 3 (DC2, DC3)	3168
74.4.5	Nexus Development Control Register 4 (DC4)	3172
74.4.6	Development Status Register (DS)	3173
74.4.7	Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers	3174
74.4.8	Nexus Watchpoint Mask (WMSK) register	3180

74.4.9	Nexus Overrun Control (OVCR) register	3181
74.4.10	Data Trace Control (DTC) register	3182
74.4.11	Data Trace Start Address Registers (DTSA1–4)	3184
74.4.12	Data Trace End Address 1–4 (DTEA1–4) registers	3185
74.4.13	Read/Write Access Control/Status (RWCS) register	3186
74.4.14	Read/Write Access Data (RWD) register	3187
74.4.15	Read/Write Access Address (RWA)	3189
74.5	Nexus 3 Register Access via JTAG/OnCE	3189
74.6	Nexus 3 Register Access via Software	3190
74.7	Nexus Message Fields	3190
74.7.1	TCODE Field	3190
74.7.2	Source ID Field (SRC)	3191
74.7.3	Relative Address Field (U-ADDR)	3191
74.7.4	Full Address Field (F-ADDR)	3192
74.8	Nexus Message Queues	3192
74.8.1	Message Queue Overrun	3193
74.8.2	CPU Stall	3193
74.8.3	Message Suppression	3193
74.8.4	Nexus Message Priority	3193
74.8.5	Data Acquisition Message Priority Loss Response	3195
74.8.6	Ownership Trace Message Priority Loss Response	3195
74.8.7	Program Trace Message Priority Loss Response	3195
74.8.8	Program Trace Sync Message Priority Loss Response	3195
74.8.9	Data Trace Message Priority Loss Response	3195
74.9	Debug Status messages	3195
74.10	Error messages	3196
74.11	Ownership trace	3196
74.11.1	Overview	3196
74.11.2	Ownership Trace Messaging (OTM)	3196
74.12	program Trace	3197
74.12.1	Branch Trace Messaging types	3198
74.12.2	BTM Message Formats	3199
74.12.3	Program Trace Message Fields	3200
74.12.4	Resource Full Messages	3201
74.12.5	Program Correlation Messages	3202
74.12.6	Program Trace Overflow Error messages	3203

74.12.7	Program Trace Synchronization messages	3203
74.12.8	Program Trace Direct/Indirect Branch with Sync messages	3205
74.12.9	Enabling Program Trace	3206
74.12.10	Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)	3206
74.13	Data Trace	3208
74.13.1	Data Trace Messaging (DTM)	3208
74.13.2	DTM Message formats	3208
74.13.3	DTM operation	3210
74.13.4	Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)	3212
74.14	Data Acquisition messaging	3212
74.14.1	Data Acquisition ID Tag field	3212
74.14.2	Data Acquisition Data field	3213
74.14.3	Data Acquisition Trace event	3213
74.15	Watchpoint Trace messaging	3213
74.15.1	Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)	3215
74.16	External Hardware Trigger Controls	3215
74.17	Nexus 3 Read/Write access to memory-mapped resources	3216
74.17.1	Single write access	3217
74.17.2	Block write access	3217
74.17.3	Single read access	3218
74.17.4	Block read access	3218
74.17.5	Error handling	3219
74.17.6	Read/Write access error message	3220
74.18	Nexus 3 pin interface	3220
74.18.1	Pins implemented	3220
74.18.2	Pin protocol	3223
74.19	Rules for output messages	3225
74.20	Auxiliary port arbitration	3226
74.21	Examples	3226
74.22	Electrical characteristics	3229
74.23	IEEE 1149.1 (JTAG) RD/WR sequences	3229
74.23.1	JTAG sequence for accessing internal Nexus registers	3229
74.23.2	JTAG sequence for read access of memory-mapped resources	3230
74.23.3	JTAG sequence for write access of memory-mapped resources	3230

75 e200z425n3 Nexus 3 Module 3231

75.1	Introduction	3231
75.1.1	General Description	3231
75.1.2	Terms and Definitions	3231
75.1.3	Feature List	3232
75.1.4	Functional Block Diagram	3234
75.2	Enabling Nexus 3 Operation	3234
75.2.1	Interaction with Low Power Modes	3235
75.3	TCODEs supported	3235
75.4	Nexus 3 Programmer's Model	3241
75.4.1	Client Select Control (CSC) register	3243
75.4.2	Port Configuration Register (PCR) — reference only	3243
75.4.3	Nexus Development Control Register 1 (DC1)	3244
75.4.4	Nexus Development Control Registers 2 & 3 (DC2, DC3)	3245
75.4.5	Nexus Development Control Register 4 (DC4)	3250
75.4.6	Development Status Register (DS)	3251
75.4.7	Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers	3252
75.4.8	Nexus Watchpoint Mask (WMSK) register	3258
75.4.9	Nexus Overrun Control (OVCR) register	3259
75.4.10	Data Trace Control (DTC) register	3260
75.4.11	Data Trace Start Address Registers (DTSA1–4)	3262
75.4.12	Data Trace End Address 1–4 (DTEA1–4) registers	3263
75.4.13	Read/Write Access Control/Status (RWCS) register	3264
75.4.14	Read/Write Access Data (RWD) register	3265
75.4.15	Read/Write Access Address (RWA)	3267
75.5	Nexus 3 Register Access via JTAG/OnCE	3267
75.6	Nexus 3 Register Access via Software	3268
75.7	Nexus Message Fields	3268
75.7.1	TCODE Field	3268
75.7.2	Source ID Field (SRC)	3269
75.7.3	Relative Address Field (U-ADDR)	3269
75.7.4	Full Address Field (F-ADDR)	3270
75.8	Nexus Message Queues	3270
75.8.1	Message Queue Overrun	3271
75.8.2	CPU Stall	3271
75.8.3	Message Suppression	3271

75.8.4	Nexus Message Priority	3271
75.8.5	Data Acquisition Message Priority Loss Response	3273
75.8.6	Ownership Trace Message Priority Loss Response	3273
75.8.7	Program Trace Message Priority Loss Response	3273
75.8.8	Program Trace Sync Message Priority Loss Response	3273
75.8.9	Data Trace Message Priority Loss Response	3273
75.9	Debug Status messages	3273
75.10	Error messages	3274
75.11	Ownership trace	3274
75.11.1	Overview	3274
75.11.2	Ownership Trace Messaging (OTM)	3274
75.12	program Trace	3275
75.12.1	Branch Trace Messaging types	3276
75.12.2	BTM Message For mats	3277
75.12.3	Program Trace Message Fields	3278
75.12.4	Resource Full Messages	3280
75.12.5	Program Correlation Messages	3281
75.12.6	Program Trace Overflow Error messages	3282
75.12.7	Program Trace Synchronization messages	3282
75.12.8	Program Trace Direct/Indirect Branch with Sync messages	3283
75.12.9	Enabling Program Trace	3285
75.12.10	Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration) ..	3285
75.13	Data Trace	3286
75.13.1	Data Trace Messaging (DTM)	3286
75.13.2	DTM Message formats	3287
75.13.3	DTM operation	3289
75.13.4	Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)	3291
75.14	Data Acquisition messaging	3291
75.14.1	Data Acquisition ID Tag field	3291
75.14.2	Data Acquisition Data field	3292
75.14.3	Data Acquisition Trace event	3292
75.15	Watchpoint Trace messaging	3292
75.15.1	Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)	3294
75.16	External Hardware Trigger Controls	3294
75.17	Nexus 3 Read/Write access to memory-mapped resources	3295
75.17.1	Single write access	3296

75.17.2	Block write access	3296
75.17.3	Single read access	3297
75.17.4	Block read access	3297
75.17.5	Error handling	3298
75.17.6	Read/Write access error message	3299
75.18	Nexus 3 pin interface	3299
75.18.1	Pins implemented	3299
75.18.2	Pin protocol	3301
75.19	Rules for output messages	3304
75.20	Auxiliary port arbitration	3304
75.21	Examples	3304
75.22	Electrical characteristics	3307
75.23	IEEE 1149.1 (JTAG) RD/WR sequences	3307
75.23.1	JTAG sequence for accessing internal Nexus registers	3307
75.23.2	JTAG sequence for read access of memory-mapped resources	3308
75.23.3	JTAG sequence for write access of memory-mapped resources	3308
76	GTM Development Interface (GTMDI)	3309
76.1	Introduction	3309
76.2	Overview	3310
76.2.1	Features	3313
76.2.2	Modes of operation	3314
76.2.3	DATA port	3314
76.3	External signal description	3314
76.3.1	Event in (EVTI)	3315
76.3.2	Event out EVTO[1:0]	3315
76.3.3	Test clock input (TCK)	3315
76.3.4	Test data input (TDI)	3315
76.3.5	Test data output (TDO)	3315
76.3.6	Test Mode Select (TMS)	3316
76.4	Register definition	3316
76.4.1	Register descriptions	3318
76.4.2	Unimplemented registers	3380
76.5	Functional description	3380
76.5.1	Debug enable logic	3381
76.5.2	TIM-TOM-ATOM selection and control logic	3381

76.5.3	GTMDI reset configuration	3383
76.5.4	Message Data Bus —interface with NAR module	3383
76.5.5	IEEE 1149.1 (JTAG) input port	3387
76.5.6	Nexus Class 1 development support	3392
76.5.7	Debug status	3392
76.5.8	Data trace	3392
76.5.9	Fetch trace	3395
76.5.10	Watchpoint trace	3397
76.6	Initialization/application information	3397
76.6.1	Accessing GTMDI tool-mapped registers	3397
77	Nexus Crossbar Multi-Master Client (NXMC)	3399
77.1	Introduction	3399
77.1.1	Features	3399
77.2	External signal description	3399
77.3	Supported messages and TCODEs	3400
77.4	Register description	3403
77.4.1	Register descriptions	3404
77.5	Programming considerations (RESET)	3411
77.5.1	IEEE 1149.1 (JTAG) Test Access Port	3411
77.5.2	Enabling NXMC operation	3411
77.5.3	Enabling NXMC TAP controller	3411
77.5.4	NXMC register access via JTAG	3412
77.6	Functional description	3412
77.6.1	Data trace	3412
77.6.2	Watchpoint support	3417
77.6.3	Peripheral interface	3418
77.6.4	Timestamping feature	3419
78	Cyclic Redundancy Check (CRC) Unit	3421
78.1	Introduction	3421
78.2	Main features	3421
78.2.1	Standard features	3421
78.3	Block diagram	3421
78.4	Signal description	3422
78.4.1	Peripheral bus interface	3422

78.5	Memory map and registers	3422
78.5.1	Register description	3422
78.6	Functional description	3426
78.7	Use cases	3428
79	Memory Error Management Unit (MEMU)	3431
79.1	Introduction	3431
79.2	Features	3431
79.3	Block diagram	3432
79.4	Design overview	3432
79.5	External signal description	3436
79.6	Memory map and register definition	3437
79.6.1	Memory map	3437
79.6.2	Register descriptions	3439
79.7	Functional description	3454
79.7.1	Initializing MEMU	3454
79.7.2	Reading the reporting table	3455
79.7.3	Handling overflows (Multiple error reporting)	3456
80	Indirect Memory Access (IMA)	3458
80.1	Introduction	3458
80.2	Accessing memory via CPU End2End ECC test	3458
80.2.1	Accessing I-cache and D-cache memory via processor core	3458
80.2.2	Accessing non I-cache or D-cache memory via processor core	3460
80.3	Accessing memory via the IMA module	3462
80.3.1	Features	3462
80.3.2	Block diagram	3462
80.3.3	IMA Module Memory Map and Registers	3463
80.3.4	IMA module functional description	3471
80.3.5	Application information	3473
81	Fault Collection and Control Unit (FCCU)	3475
81.1	Introduction	3475
81.1.1	Acronyms and abbreviations	3476
81.2	Main features	3476
81.2.1	Standard features	3476

81.3	Block diagram	3477
81.4	Signal description	3478
81.4.1	Pinout	3478
81.4.2	IPS bus interface	3478
81.5	Functional description	3478
81.5.1	Definitions	3478
81.5.2	FSM description	3479
81.5.3	Reset interface	3480
81.5.4	Fault priority scheme and nesting	3481
81.5.5	Fault recovery	3482
81.5.6	NMI interface	3484
81.5.7	FAULT interface	3484
81.5.8	STCU interface	3485
81.5.9	EOUT interface	3486
81.5.10	Error signal flow	3492
81.6	Register description	3493
81.6.1	FCCU Control Register (FCCU_CTRL)	3496
81.6.2	FCCU CTRL Key Register (FCCU_CTRLK)	3498
81.6.3	FCCU Configuration Register (FCCU_CFG)	3498
81.6.4	FCCU RF Configuration Registers (FCCU_RF_CFG[0:3])	3500
81.6.5	FCCU RFS Configuration Registers (FCCU_RFS_CFG[0:7])	3502
81.6.6	FCCU RF Status Register (FCCU_RF_S[0:3])	3503
81.6.7	FCCU RF Key Register (FCCU_RFK)	3505
81.6.8	FCCU RF Enable Registers (FCCU_RF_E[0:3])	3505
81.6.9	FCCU RF Timeout Enable Registers (FCCU_RF_TOE[0:3])	3506
81.6.10	FCCU RF Time-out Register (FCCU_RF_TO)	3507
81.6.11	FCCU CFG Timeout Register (FCCU_CFG_TO)	3508
81.6.12	FCCU IO Control Register (FCCU_EINOUT)	3509
81.6.13	FCCU Status Register (FCCU_STAT)	3510
81.6.14	FCCU NA Freeze Status Register (N2AF_STATUS)	3512
81.6.15	FCCU AF Freeze Status Register (A2FF_STATUS)	3513
81.6.16	FCCU NF Freeze Status Register (N2FF_STATUS)	3514
81.6.17	FCCU FA Freeze Status Register (F2A_STATUS)	3515
81.6.18	FCCU RF Fake Register (FCCU_RFF)	3516
81.6.19	FCCU IRQ Status Register (FCCU_IRQ_STAT)	3517
81.6.20	FCCU IRQ Enable Register (FCCU_IRQ_EN)	3518
81.6.21	FCCU XTMR Register (FCCU_XTMR)	3519

81.6.22	FCCU MCS Register (FCCU_MCS)	3520
81.6.23	FCCU Transient Register (FCCU_TRANS_LOCK)	3521
81.6.24	FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)	3522
81.6.25	FCCU Delta T Register (FCCU_DELTA_T)	3522
81.6.26	FCCU IRQ Alarm Enable Registers (FCCU_IRQ_ALARM_EN[0:3])	3523
81.6.27	FCCU NMI Enable Registers (FCCU_NMI_EN[0:3])	3524
81.6.28	FCCU EOUT Signaling Enable Registers (FCCU_EOUT_SIG_EN[0:3])	3524
81.7	Register map	3526
81.8	FCCU Output Supervision Unit (FOSU)	3529
81.9	Use cases and limitations	3530
81.9.1	Misconfigurations	3530
81.9.2	Recommendations to configure FCCU	3531
82	Self-Test Control Unit (STCU2)	3532
82.1	Introduction	3532
82.2	Main features	3533
82.3	Block diagram	3534
82.3.1	IPS bus interface	3535
82.4	Functional description	3536
82.4.1	FSM description	3536
82.4.2	Reset management	3536
82.4.3	LBIST and MBIST scheduling	3537
82.4.4	ABORT management	3537
82.4.5	PLL interface	3538
82.4.6	Interrupt interface	3538
82.4.7	FCCU interface	3538
82.4.8	Watchdogs	3538
82.4.9	CRC	3539
82.4.10	SSCM interface	3539
82.5	Register description	3540
82.5.1	Register description conventions	3541
82.5.2	STCU2 Run Register (STCU_RUN)	3542
82.5.3	STCU2 Run Software Register (STCU_RUNSW)	3543
82.5.4	STCU2 SK Code Register (STCU_SKC)	3544
82.5.5	STCU2 Configuration Register (STCU_CFG)	3546

82.5.6	STCU2 PLL Configuration Register (STCU_PLL_CFG)	3548
82.5.7	STCU2 Watchdog Register Granularity (STCU_WDG)	3549
82.5.8	STCU2 Interrupt Flag Register (STCU_INT_FLG)	3550
82.5.9	STCU2 CRC Expected Status Register (STCU_CRCE)	3551
82.5.10	STCU2 CRC Read Status Register (STCU_CRCR)	3552
82.5.11	STCU2 Error Register (STCU_ERR_STAT)	3552
82.5.12	STCU2 Error FM Register (STCU_ERR_FM)	3555
82.5.13	STCU2 Off-Line LBIST Status Register (STCU_LBS)	3556
82.5.14	STCU2 Off-Line LBIST End Flag Register (STCU_LBE)	3557
82.5.15	STCU2 On-Line LBIST Status Register (STCU_LBSSW)	3558
82.5.16	STCU2 On-Line LBIST End Flag Register (STCU_LBESW)	3559
82.5.17	STCU2 On-Line LBIST Reset Management (STCU_LBRMSW)	3559
82.5.18	STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)	3560
82.5.19	STCU2 Off-Line MBIST Status Low Register (STCU_MBSL)	3561
82.5.20	STCU2 Off-Line MBIST Status Medium Register (STCU_MBSM)	3562
82.5.21	STCU2 Off-Line MBIST Status High Register (STCU_MBSH)	3562
82.5.22	STCU2 Off-Line MBIST End Flag Low Register (STCU_MBEL)	3563
82.5.23	STCU2 Off-Line MBIST End Flag Medium Register (STCU_MBEM)	3564
82.5.24	STCU2 Off-Line MBIST End Flag High Register (STCU_MBEH)	3565
82.5.25	STCU2 On-Line MBIST Status Low Register (STCU_MBSLSW)	3565
82.5.26	STCU2 On-Line MBIST Status Medium Register (STCU_MBSMSW)	3566
82.5.27	STCU2 On-Line MBIST Status High Register (STCU_MBSHSW)	3567
82.5.28	STCU2 On-Line MBIST End Flag Low Register (STCU_MBELSW)	3568
82.5.29	STCU2 On-Line MBIST End Flag Medium Register (STCU_MBEMSW)	3569
82.5.30	STCU2 On-Line MBIST End Flag High Register (STCU_MBEHSW)	3570
82.5.31	STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)	3571
82.5.32	STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)	3572
82.5.33	STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)	3573
82.5.34	STCU2 LBIST Control Register (STCU_LB_CTRL)	3574
82.5.35	STCU2 LBIST PC Stop Register (STCU_LB_PCS)	3576
82.5.36	STCU2 LBIST PRPG Low Register (STCU_LB_PRPGL)	3577
82.5.37	STCU2 LBIST PRPG High Register (STCU_LB_PRPGH)	3578
82.5.38	STCU2 Off-Line LBIST MISR Expected Low Register (STCU_LB_MISREL)	3579

82.5.39	STCU2 Off-Line LBIST MISR Expected High Register (STCU_LB_MISREH)	3580
82.5.40	STCU2 Off-Line LBIST MISR Read Low Register (STCU_LB_MISRRL)	3581
82.5.41	STCU2 Off-Line LBIST MISR Read High Register (STCU_LB_MISRRLH)	3581
82.5.42	STCU2 On-Line LBIST MISR Expected Low Register (STCU_LB_MISRELSW)	3582
82.5.43	STCU2 On-Line LBIST MISR Expected High Register (STCU_LB_MISREHSW)	3583
82.5.44	STCU2 On-Line LBIST MISR Read Low Register (STCU_LB_MISRRLSW)	3584
82.5.45	STCU2 On-Line LBIST MISR Read High Register (STCU_LB_MISRRLHSW)	3585
82.5.46	STCU2 MBIST Control Register (STCU_MB_CTRL)	3586
82.6	Register map	3588
82.7	Use cases and limitations	3593
82.7.1	Off-Line Self-Test Sequence	3593
82.7.2	On-Line Self-Test Sequence	3594
82.7.3	Bypass USER Mode	3596
82.7.4	Design implementation information	3597
82.8	Open issues	3597
83	Register Protection (REG_PROT)	3598
83.1	Introduction	3598
83.1.1	Overview	3598
83.1.2	Features	3598
83.1.3	Modes of operation	3599
83.2	External signal description	3599
83.3	Memory map and register definition	3599
83.3.1	Memory map	3600
83.3.2	Register descriptions	3601
83.4	Functional description	3603
83.4.1	General	3603
83.4.2	Change lock settings	3604
83.4.3	Access errors	3606
83.5	Initialization/application information	3607
83.5.1	Reset	3607

	83.5.2	Writing C code using the register protection scheme	3607
84		Password and Device Security Module (PASS)	3609
	84.1	Introduction	3609
	84.1.1	Overview	3609
	84.1.2	Features	3609
	84.1.3	Modes of operation	3609
	84.2	External signal description	3610
	84.3	Memory map/register definition	3610
	84.3.1	Register descriptions	3611
	84.4	Functional description	3625
	84.4.1	Life cycle and security	3625
	84.5	Initialization/application information	3626
	84.5.1	Reset	3626
	84.5.2	Modifying life cycle (maturing the device)	3626
	84.5.3	Setting lock bits in a password group	3627
	84.5.4	Censoring/uncensoring the device	3629
	84.5.5	Unlocking the debug interface via JTAG	3629
	84.5.6	Production disable	3629
85		Tamper Detection Module (TDM)	3631
	85.1	Overview	3631
	85.2	Features	3632
	85.3	External signal description	3632
	85.4	Memory map and register description	3632
	85.4.1	DCF client	3632
	85.4.2	Memory Mapped Registers	3633
	85.4.3	DCF client and register descriptions	3633
	85.5	Functional description	3643
	85.5.1	Flash erase counter and tamper detection	3644
	85.5.2	Assigning blocks to Tamper Detect Regions (TDR)	3645
	85.5.3	Diary	3645
	85.5.4	Specifying the diary base address	3646
	85.5.5	Diary Base Address Verification	3647
	85.5.6	One Time Programmable (OTP)	3647

86	Arithmetic Modeling Unit (AMU)	3648
86.1	Introduction	3648
86.1.1	Features	3649
86.2	Memory map and register definition	3649
86.2.1	AMU memory map	3649
86.2.2	Register Descriptions	3651
86.3	Functional description	3669
86.3.1	Register data types	3671
86.3.2	Calculation precision—Deviation from IEEE standard	3671
86.3.3	DMA	3671
86.3.4	Advanced features	3672
86.4	Programming examples	3673
86.4.1	AMU operation with U, p1, p2, p3, V vector resident in local RAM	3673
86.4.2	AMU operation with p1, p2, p3, U vector in local RAM, and V vector in flash	3674
86.4.3	AMU resume	3674
86.4.4	DMA only operation	3674
87	Wakeup Unit (WKPU)	3676
87.1	Introduction	3676
87.1.1	Overview	3676
87.1.2	Features	3677
87.2	External signal description	3677
87.3	Memory map and register description	3677
87.3.1	Memory map	3677
87.3.2	Register descriptions	3677
87.4	Functional description	3682
87.4.1	Non-maskable interrupts	3682
87.4.2	Reset management	3682
87.4.3	System wakeup	3682
87.4.4	Glitch filter	3682
87.5	Initialization information	3683
87.5.1	Glitch filter and pad configuration	3683
87.5.2	Non-maskable interrupt initialization	3683
87.5.3	Reset request	3684
88	GTM103 Integration (GTMINT) Module	3685

88.1	Introduction	3685
88.1.1	Features	3685
88.1.2	Modes of Operation	3687
88.1.3	Block Diagram	3687
88.2	Memory Map and Register Definition	3688
88.2.1	Module Memory Map	3688
88.2.2	Register Descriptions	3709
88.3	Functional Description	3713
88.3.1	GTM-IP Timer	3714
88.3.2	GTMDI Debug Interface	3715
88.3.3	Error Detection on GTM Memories Accesses	3716
88.3.4	Stop Mode Description	3719
88.3.5	Debug / Halt Mode Description	3719
88.3.6	GTM-IP Timer Software Reset Description	3720
88.3.7	AEI Interface Software Reset Description	3720
88.3.8	Interrupt and DMA Support Logic	3720
88.4	Initialization/Application Information	3721
89	Emulation and Debug Device Introduction	3722
89.1	Functionality	3722
89.2	Feature summary	3722
89.3	Block diagram	3723
89.4	Clocking	3725
89.5	Reset logic	3727
89.6	BD memory map	3728
90	Buddy Device—Crossbar and RAM (BD_XBS_RAM)	3730
90.1	3 x 5 Crossbar switch and attached controllers	3730
90.2	BD_XBS_RAM End-to-End ECC32 Checkbit/Syndrome Coding Scheme	3731
91	Buddy Device—System Integration Unit Lite (BD_SIUL2)	3738
91.1	Introduction	3738
91.2	Device identification	3738
91.3	Device status information	3739
91.3.1	Initialization status	3739

91.3.2	PD reset status	3739
91.4	Memory map	3739
91.4.1	Register descriptions	3740
92	Buddy Device—Debug and Calibration Interface (DCI)	3744
92.1	Introduction	3744
92.1.1	Features	3744
92.1.2	Overview	3744
92.1.3	Operating modes	3747
92.2	External signal description	3760
92.3	Register description	3760
92.3.1	DCI control register (DCI_CR)	3761
92.3.2	DCI EVTx pin multiplexing control register (DCI_PINCR)	3762
92.4	Functional description	3762
92.4.1	DCI mode transition	3762
92.4.2	LFAST LVDS pad fault	3764
93	Buddy Device—JTAG Controller (JTAGC)	3766
93.1	Introduction	3766
93.1.1	Block Diagram	3766
93.1.2	Features	3767
93.1.3	Modes of operation	3767
93.2	External signal description	3768
93.2.1	TCK—Test clock input	3768
93.2.2	TDI—Test data input	3768
93.2.3	TDO—Test data output	3768
93.2.4	TMS—Test mode select	3768
93.2.5	JCOMP—JTAG compliancy	3768
93.3	Register description	3769
93.3.1	Instruction register	3769
93.3.2	Bypass register	3769
93.3.3	Device identification register	3769
93.3.4	JTAG_PASSWORD register	3770
93.3.5	Boundary scan register	3771
93.4	Functional description	3771
93.4.1	JTAGC reset configuration	3771

	93.4.2	IEEE 1149.1-2001 (JTAG) Test Access Port	3771
	93.4.3	TAP controller state machine	3771
	93.4.4	JTAGC block instructions	3773
	93.4.5	Boundary scan	3776
	93.5	Initialization/Application information	3776
94		Buddy Device—JTAG Master (JTAGM)	3778
	94.1	Introduction	3778
	94.1.1	Overview	3778
	94.1.2	Feature description	3779
	94.2	Functional description	3779
	94.2.1	Module architecture	3779
	94.2.2	JTAGM JCOMP usage in LFAST	3780
	94.2.3	JTAGM data error detection	3781
	94.2.4	JTAGM message tagging	3781
	94.2.5	JTAGM Ready signal	3781
	94.2.6	EVTO and EVTI signals	3782
	94.2.7	JTAGM configuration and status monitoring	3782
	94.2.8	JTAGM to DCI serial interface	3783
	94.2.9	JTAGM data handling and CRC calculation in LFAST mode	3783
	94.2.10	Software interface	3784
	94.3	Modes of operation	3785
	94.4	Memory map and register definition	3785
	94.4.1	Register descriptions	3785
95		Buddy Device—Nexus Aurora Router (NAR)	3794
	95.1	Introduction	3794
	95.2	Overview	3795
	95.3	Features	3795
	95.4	Register definition	3796
	95.4.1	Register descriptions	3797
	95.4.2	Message filtering registers	3800
	95.4.3	Trace memory bus configuration registers	3803
	95.4.4	Suppress mode triggers register (NAR_STCR)	3806
	95.4.5	Receive queue client disable register (NAR_CDR)	3808
	95.4.6	Nexus Aurora Phy control register (NAR_NAPCR)	3810

95.4.7	Trace Memory fill level and partition configuration register (NAR_AHFPAR)	3810
95.5	Functional description	3812
95.5.1	Reset/Startup	3812
95.5.2	Operation modes	3813
95.5.3	Data reception and arbitration	3814
95.5.4	Suppress mode	3816
95.5.5	Stall detection	3817
95.5.6	Output arbitration	3817
95.5.7	Queue partitioning	3818
95.5.8	Message translation and formatting	3819
95.5.9	Aurora interface	3820
95.5.10	Trace memory bus output port usage	3820
95.5.11	Internal message generation	3822
95.5.12	Timestamp function	3825
95.5.13	Global (Development tool client) stall control	3826
95.5.14	Overlay and internal trace memory full status generation	3827
95.5.15	Configuration/debug interface JTAG (1149.1)	3828
95.5.16	Trace memory bus controller	3830
95.5.17	Aurora client controller	3832
95.6	Initialization information	3832
95.7	Application information	3833
96	Buddy Device—Nexus Aurora Link (NAL)	3834
96.1	Transmit operation	3835
96.2	Nexus Aurora formatting	3835
96.3	Static training	3836
96.4	Programmable registers	3836
96.4.1	Nexus Aurora Link General Status Register (NAL_GSR)	3837
96.4.2	NAL General Control Register (NAL_GCR)	3837
96.4.3	NAL Training Control Register (NAL_TCR)	3839
97	Buddy Device—Nexus Aurora PHY (NAP)	3842
97.1	Introduction	3842
97.1.1	Features	3843
97.1.2	Modes of operation	3843
97.2	External signal description	3844

97.3	Memory map and register definition	3844
97.4	Functional description	3844
97.4.1	Data symbol character accumulator	3844
97.4.2	Serializer	3844
97.4.3	Clock generation and control	3845
97.5	Initialization information	3845
98	Buddy Device—Data Write Processing Unit (DWPU)	3846
98.1	Introduction	3846
98.2	Features	3847
98.3	Functional description	3847
98.3.1	Nexus Out Interface (NOI)	3847
98.3.2	Message Concatenation and Allocation (MCA)	3848
98.3.3	MCA has the capability of providing one fully concatenated message to a CSPU each clock.Client Specific Processing (CSP)	3848
98.3.4	Address Tag Check (ATC)	3851
98.3.5	Out of Range Check (ORC)	3853
98.3.6	Message Formatter (MF)	3855
98.3.7	Nexus Client Interface (NCI)	3856
98.3.8	IEEE 1149.1 (JTAG) Test Access Port	3856
98.3.9	Enabling DWPU operation	3856
98.3.10	Enabling TAP controller	3856
98.3.11	Register access via JTAG	3857
98.4	Configuration	3858
98.4.1	Tag RAM initialization	3858
98.5	Register definition	3859
98.5.1	Register descriptions	3861
99	Buddy Device—Nexus Read Write Access (RWA)	3872
99.1	Register definition	3872
99.1.1	Register descriptions	3872
99.1.2	Read/write access data (RWD)	3875
99.1.3	Read/write access address (RWA)	3876
99.2	Nexus register access via JTAG	3876
99.3	Memory access examples	3877
99.3.1	Single write access	3877

	99.3.2	Block write access	3877
	99.3.3	Single read access	3878
	99.3.4	Block read access	3878
100		Buddy Device—LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed Debug	3879
	100.1	Introduction	3879
	100.2	Block diagram	3879
	100.3	External signals	3880
	100.3.1	LFAST operating data rates	3880
	100.4	LFAST frame structure	3880
	100.5	Features	3883
	100.6	Memory map and register definition	3883
	100.6.1	Memory map	3883
	100.6.2	Register descriptions	3886
	100.6.3	Register safety classification requirements	3910
	100.7	Functional description	3910
	100.7.1	Startup procedure	3910
	100.7.2	LFAST Interface enable signal (lfast_sysclk_en)	3911
	100.7.3	Line Receiver (LR)	3912
	100.7.4	Transmit Controller	3921
	100.7.5	Frames supported	3925
	100.7.6	Frame flow	3925
	100.7.7	Test and Debug Support	3928
	100.8	Packet memory	3936
	100.9	Resets	3936
	100.10	Clocks	3937
	100.10.1	Clocking strategy	3938
	100.10.2	Slow speed clock	3938
	100.10.3	Rx controller clocks	3941
	100.10.4	Tx controller clocks	3942
101		Revision History	3944

List of tables

Table 1.	Register Conventions	154
Table 2.	Acronyms and abbreviations	155
Table 3.	Processor cores	157
Table 4.	Features List	161
Table 5.	ECC comparison of data write-then-read sequence	173
Table 6.	Pin/ball startup and reset states	177
Table 7.	Power supply and reference pins	178
Table 8.	System pins	179
Table 9.	LVDS pin descriptions	179
Table 10.	Aurora pin descriptions	183
Table 11.	Overview Memory Map	185
Table 12.	Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map	187
Table 13.	Flash memory and overlay RAM map	194
Table 14.	Buddy device memory map	197
Table 15.	RAM memory map	198
Table 16.	UTest flash memory map	198
Table 17.	PMHF module categories	202
Table 18.	Module classification	203
Table 19.	LBIST partitions	209
Table 20.	Not in LBIST partitions	212
Table 21.	FCCU failure inputs	215
Table 22.	ECC RAM implementations	229
Table 23.	Reset settings for e200z4 resources	233
Table 24.	Special Purpose Registers (SPRs)	236
Table 25.	Reference links to related information	240
Table 26.	Computational shell crossbar switch master assignments	241
Table 27.	Computational shell crossbar switch slave assignments	241
Table 28.	Flash memory port assignments	241
Table 29.	Reference links to related information	242
Table 30.	Peripheral shell crossbar switch master assignments	243
Table 31.	Peripheral shell crossbar switch slave assignments	243
Table 32.	Reference links to related information	244
Table 33.	SMPU0 logical bus master assignments	244
Table 34.	Reference links to related information	246
Table 35.	SMPU1 logical bus master assignments	246
Table 36.	Reference links to related information	248
Table 37.	IAHB_BE1 Register field description	250
Table 38.	IAHB_BE2 field description	252
Table 39.	INTC implemented registers	253
Table 40.	Interrupt sources	254
Table 41.	Reference links to related information	295
Table 42.	SPC57EM80xx multiplexer and channel list	295
Table 43.	DMAMUX peripheral DMA request to input source mapping	298
Table 44.	PIT_0 periodic trigger to DMA channel mapping	300
Table 45.	DMAMUX memory map	300
Table 46.	Reference links to related information	300
Table 47.	Reference links to related information	301
Table 48.	Reference links to related information	301

Table 49.	Flash controller bus master assignments	302
Table 50.	Calibration memory access times	302
Table 51.	Minimum wait state settings for overlay flash latency match	303
Table 52.	Analog modules	303
Table 53.	Timers	305
Table 54.	System Timer Module (STM) instances	306
Table 55.	Software Watchdog Timer (SWT) instances	306
Table 56.	Register reset values	307
Table 57.	Reference links to related information	309
Table 58.	PIT Information	309
Table 59.	PIT0 channel assignments for periodic DMA triggering	309
Table 60.	PIT0 memory map	310
Table 61.	PIT1 memory map	310
Table 62.	FEC interface selection	311
Table 63.	CAN bus controllers	311
Table 64.	Reference text to related information	312
Table 65.	Reference links to related information	313
Table 66.	DSPI instantiation	314
Table 67.	DSPI registers memory map	315
Table 68.	Flexray pin names	316
Table 69.	PSI5 controller generics	317
Table 70.	SENT channel configuration	318
Table 71.	SENT Receiver memory map	318
Table 72.	LINFlexD configurations	320
Table 73.	SSCM_MEMCONFIG Reset Values	321
Table 74.	SSCM startup information addresses	321
Table 75.	Reference text to related information	322
Table 76.	CRC Contexts	322
Table 77.	MEMU error sources	323
Table 78.	Number of entries in each reporting table	324
Table 79.	Corresponding error sources for System RAM OFLW0 register	324
Table 80.	Corresponding error sources for System RAM OFLW1 register	325
Table 81.	Corresponding error sources for System RAM OFLW2 register	326
Table 82.	Corresponding error sources for Peripheral RAM OFLW register	327
Table 83.	Corresponding error sources for FLASH OFLW register	328
Table 84.	MEMU registers	329
Table 85.	SRAM with processor Core access	332
Table 86.	SRAM arrays accessible via IMA	333
Table 87.	No ECC bit access	334
Table 88.	Availability and reset values of selected FCCU registers	335
Table 89.	Reference links to related information	336
Table 90.	Protected SIUL2 registers	337
Table 91.	Protected CMU registers	339
Table 92.	Protected CMU registers	339
Table 93.	Protected PMC_DIG registers	339
Table 94.	Protected PLL_DIG registers	340
Table 95.	Protected OSC_DIG registers	341
Table 96.	Protected MEMU registers	341
Table 97.	Protected MC_ME registers	341
Table 98.	Protected IMA registers	342
Table 99.	Protected MC_CGM registers	342
Table 100.	Protected MC_RGM registers	342

Table 101.	Protected SSCM registers	343
Table 102.	Protected IMA registers	343
Table 103.	Protected JTAGM registers	343
Table 104.	Reference links to related information	343
Table 105.	PASS DCF records	349
Table 106.	TO[TOEn] field mapping	351
Table 107.	Tamper Detection Module (TDM) DCF clients	355
Table 108.	Debug and calibration features per SPC57EM80xx type	356
Table 109.	Module status during reset phases	364
Table 110.	DCF record field description	393
Table 111.	Series of DCF records in UTEST flash memory	395
Table 112.	DCF client list	399
Table 113.	UTEST Miscellaneous DCF Client field descriptions	415
Table 114.	PMC_REE bits DCF client field descriptions	416
Table 115.	SYSTEM modes	419
Table 116.	USER modes	420
Table 117.	Device power management controller external signals	422
Table 118.	POR and voltage monitors description	429
Table 119.	Voltage monitors configurability	431
Table 120.	Functional terminal state during power-up and reset	435
Table 121.	Reference links to related information	446
Table 122.	PD DCI signals	446
Table 123.	BD DCI signals	447
Table 124.	DCI_PINCR register field descriptions	448
Table 125.	L1SEL0 register field descriptions	450
Table 126.	L1SEL1 register field descriptions	451
Table 127.	L1SEL2 register field descriptions	453
Table 128.	L1SEL3 register field descriptions	454
Table 129.	L1SEL4 register field descriptions	456
Table 130.	L1SEL5 register field descriptions	457
Table 131.	L1SEL6 register field descriptions	459
Table 132.	L1SEL7 register field descriptions	460
Table 133.	Reference links to related information	462
Table 134.	Nexus message types	463
Table 135.	Nexus client JTAG enable instructions	464
Table 136.	Reference links to related information	466
Table 137.	PD NAR clients	468
Table 138.	BD NAR clients	469
Table 139.	NAR_NAPCR field descriptions	470
Table 140.	Reference links to related information	471
Table 141.	NXMC source and master client IDs	473
Table 142.	Reference links to related information	474
Table 143.	Concurrent instruction issue capabilities for e200z420n3	488
Table 144.	EFPU Status and Control Register field descriptions	489
Table 145.	L1CSR0 field descriptions	492
Table 146.	L1CSR1 field descriptions	494
Table 147.	L1CFG0 field descriptions	496
Table 148.	L1CFG1 field descriptions	497
Table 149.	L1FINV0 field descriptions	499
Table 150.	L1FINV1 field descriptions	500
Table 151.	Exceptions and conditions	502
Table 152.	ESR field descriptions	503

Table 153.	MSR field descriptions	505
Table 154.	Machine Check Syndrome Register (MCSR) field descriptions	506
Table 155.	IVPR field descriptions	509
Table 156.	Critical Input Interrupt—register settings	509
Table 157.	Machine Check Interrupt—register settings	510
Table 158.	Data Storage Interrupt—register settings	510
Table 159.	Instruction Storage Interrupt—register settings	511
Table 160.	External Input Interrupt—register settings	511
Table 161.	Alignment Interrupt—register settings	512
Table 162.	Program Interrupt—register settings	513
Table 163.	System Call Interrupt—register settings	514
Table 164.	Debug Interrupt—register settings	514
Table 165.	System Reset Interrupt—register settings	515
Table 166.	Embedded Floating-point Data Interrupt—register settings	516
Table 167.	Embedded Floating-point Round Interrupt—register settings	516
Table 168.	Performance Monitor Interrupt—Register Settings	517
Table 169.	MMUCFG field descriptions	520
Table 170.	MPU0CFG field descriptions	521
Table 171.	MPU0CSR0 field descriptions	522
Table 172.	MAS0—MPU Read/Write and Replacement Control	524
Table 173.	MAS1—descriptor context and configuration control	527
Table 174.	MAS2—Upper Bound Control	528
Table 175.	MAS3—Lower Bound Control	529
Table 176.	DMEMCFG0 field descriptions	530
Table 177.	DMEMCTL0 field descriptions	531
Table 178.	DMEMCTL1 field descriptions	534
Table 179.	IMEMCFG0 field descriptions	539
Table 180.	IMEMCTL0 field descriptions	540
Table 181.	IMEMCTL1 field descriptions	542
Table 182.	E2ECTL0 field descriptions	544
Table 183.	E2EECSR0 field descriptions	545
Table 184.	Concurrent instruction issue capabilities for e200z425n3	551
Table 185.	LSP/EFPU Status and Control Register field descriptions	552
Table 186.	L1CSR0 field descriptions	555
Table 187.	L1CSR1 field descriptions	555
Table 188.	L1CFG1 field descriptions	558
Table 189.	L1FINV1 field descriptions	559
Table 190.	Exceptions and conditions	561
Table 191.	ESR field descriptions	562
Table 192.	MSR field descriptions	563
Table 193.	Machine Check Syndrome Register (MCSR) field descriptions	565
Table 194.	IVPR field descriptions	568
Table 195.	Critical Input Interrupt—register settings	568
Table 196.	Machine Check Interrupt—register settings	569
Table 197.	Data Storage Interrupt—register settings	569
Table 198.	Instruction Storage Interrupt—register settings	570
Table 199.	External Input Interrupt—register settings	570
Table 200.	Alignment Interrupt—register settings	571
Table 201.	Program Interrupt—register settings	572
Table 202.	System Call Interrupt—register settings	573
Table 203.	Debug Interrupt—register settings	573
Table 204.	System Reset Interrupt—register settings	574

Table 205.	Embedded Floating-point Data Interrupt—register settings	575
Table 206.	Embedded Floating-point Round Interrupt—register settings	575
Table 207.	Performance Monitor Interrupt—Register Settings	576
Table 208.	MMUCFG field descriptions	579
Table 209.	MPU0CFG field descriptions	580
Table 210.	MPU0CSR0 field descriptions	581
Table 211.	MAS0—MPU Read/Write and Replacement Control	583
Table 212.	MAS1—descriptor context and configuration control	586
Table 213.	MAS2—Upper Bound Control	587
Table 214.	MAS3—Lower Bound Control	588
Table 215.	DMEMCFG0 field descriptions	589
Table 216.	DMEMCTL0 field descriptions	590
Table 217.	DMEMCTL1 field descriptions	593
Table 218.	IMEMCFG0 field descriptions	598
Table 219.	IMEMCTL0 field descriptions	599
Table 220.	IMEMCTL1 field descriptions	601
Table 221.	E2ECTL0 field descriptions	603
Table 222.	E2EECSR0 field descriptions	604
Table 223.	SIUL2 memory map	609
Table 224.	SIUL2_MIDR1 field descriptions	611
Table 225.	SIUL2_MIDR2 field description	612
Table 226.	SIUL2_DISR0 field descriptions	614
Table 227.	SIUL2_DIRER0 field descriptions	615
Table 228.	SIUL2_DIRSR0 field descriptions	616
Table 229.	SIUL2_IREER0 field descriptions	617
Table 230.	SIUL2_IFEER0 field descriptions	618
Table 231.	SIUL2_IFER0 field descriptions	619
Table 232.	SIUL2_IFMCR0 field descriptions	619
Table 233.	SIUL2_IFCPR field descriptions	620
Table 234.	SIUL2_SCR0 field descriptions	620
Table 235.	SIUL2_MSCR0 - SIUL2_MSCR511 field description	622
Table 236.	I/O MSCR Reset State Exceptions	624
Table 237.	SIUL2_MSCR512 - SIUL2_MSCR1023 field description	625
Table 238.	SIUL2_GPDO0_3 field description	626
Table 239.	SIUL2_GPDIO_3 field description	627
Table 240.	SIUL2_PGPDO0 field description	628
Table 241.	SIUL2_PGPDIO field description	628
Table 242.	SIUL2_MPGPDO0 field description	629
Table 243.	Interrupt source mapping to SIUL2 interrupt request output	634
Table 244.	XBARx memory map	636
Table 245.	XBARx_PRSn field descriptions	638
Table 246.	XBARx_CRSn field descriptions	640
Table 247.	XBIC memory map	647
Table 248.	XBIC_MCR field descriptions	648
Table 249.	XBIC_EIR field descriptions	648
Table 250.	XBIC_ESR field descriptions	649
Table 251.	XBIC_EAR Register field descriptions	650
Table 252.	Hexadecimal attribute single-bit error syndromes	651
Table 253.	XBIC H Matrix Definition	652
Table 254.	XBARx memory map	654
Table 255.	CLC_GEN field description	655
Table 257.	CLC_STGT field description	658

Table 258.	Peripheral bridge memory map	662
Table 259.	MPROT n field descriptions	665
Table 260.	PACR memory map	665
Table 261.	PACR n field descriptions	666
Table 262.	OPACR memory map	667
Table 263.	OPACR n field descriptions	669
Table 264.	SMPU x memory map	672
Table 265.	SMPU x _CESR0 field descriptions	677
Table 266.	SMPU x _CESR1 field descriptions	678
Table 267.	SMPU x _EARN field descriptions	679
Table 268.	SMPU x _EDR n field descriptions	680
Table 269.	SMPU x _RGD n _WORD0 field descriptions	681
Table 270.	SMPU x _RGD n _WORD1 field descriptions	682
Table 271.	SMPU x _RGD n _WORD2_FMT0 field descriptions	683
Table 272.	SMPU x _RGD n _WORD3 field descriptions	684
Table 273.	Protection violation definition	686
Table 274.	Overlapping region descriptor example	688
Table 275.	Operation modes available	689
Table 276.	SEMA4_GATE field descriptions	705
Table 277.	RSTGT field descriptions	706
Table 278.	SEMA4_GATE n state transitions	707
Table 279.	INTC memory map	712
Table 280.	INTC_BCR field descriptions	714
Table 281.	INTC_MPROT field descriptions	715
Table 282.	INTC_CPR n field descriptions	716
Table 283.	INTC_IACKR n field descriptions	717
Table 284.	INTC_EOIR n field descriptions	718
Table 285.	INTC_SSCIR n field descriptions	719
Table 286.	INTC_PSR n field descriptions	720
Table 287.	Selected Processor for Interrupt Request	721
Table 288.	Order of ISR execution example	730
Table 289.	Raised priority preserved events	733
Table 290.	DMA memory map	739
Table 291.	DMA_CR field descriptions	784
Table 292.	DMA_ES field descriptions	786
Table 293.	DMA_ERQH field descriptions	788
Table 294.	DMA_ERQL field descriptions	791
Table 295.	DMA_EEIH field descriptions	794
Table 296.	DMA_EEIL field descriptions	797
Table 297.	DMA_SERQ field descriptions	800
Table 298.	DMA_CERQ field descriptions	800
Table 299.	DMA_SEEI field descriptions	801
Table 300.	DMA_CEEI field descriptions	802
Table 301.	DMA_CINT field descriptions	802
Table 302.	DMA_CERR field descriptions	803
Table 303.	DMA_SSRT field descriptions	804
Table 304.	DMA_CDNE field descriptions	804
Table 305.	DMA_INTH field descriptions	806
Table 306.	DMA_INTL field descriptions	809
Table 307.	DMA_ERRH field descriptions	812
Table 308.	DMA_ERRL field descriptions	815
Table 309.	DMA_HRSH field descriptions	818

Table 310.	DMA_HRSL field descriptions	821
Table 311.	DMA_DCHPRIn field descriptions	824
Table 312.	DMA_DCHMIDn field descriptions	825
Table 313.	DMA_TCDn_SADDR field descriptions	826
Table 314.	DMA_TCDn_ATTR field descriptions	826
Table 315.	DMA_TCDn_SOFF field descriptions	827
Table 316.	DMA_TCDn_NBYTES_MLNO field descriptions	828
Table 317.	DMA_TCDn_NBYTES_MLOFFYES field descriptions	830
Table 318.	DMA_TCDn_SLAST field descriptions	831
Table 319.	DMA_TCDn_DADDR field descriptions	831
Table 320.	DMA_TCDn_CITER_ELINKYES field descriptions	832
Table 321.	DMA_TCDn_CITER_ELINKNO field descriptions	833
Table 322.	DMA_TCDn_DOFF field descriptions	834
Table 323.	DMA_TCDn_DLASTSGA field descriptions	834
Table 324.	DMA_TCDn_BITER_ELINKYES field descriptions	835
Table 325.	DMA_TCDn_BITER_ELINKNO field descriptions	836
Table 326.	DMA_TCDn_CSR field descriptions	837
Table 327.	eDMA peak transfer rates (Mbyte/s)	846
Table 328.	eDMA peak request rate (MReq/s)	847
Table 329.	TCD control and status fields	849
Table 330.	Modulo feature example	856
Table 331.	TCD status sequence – software activated channel	856
Table 332.	TCD status sequence – hardware activated channel	857
Table 333.	Channel linking parameters	858
Table 334.	Coherency model for a dynamic channel link request	859
Table 335.	Coherency model for method 1	860
Table 336.	Coherency model for method 2	861
Table 337.	Module memory map	863
Table 338.	CHCONFIGxx Field Descriptions	864
Table 339.	Channel and Trigger Enabling	864
Table 340.	MC_CGM relationship to clocks	876
Table 341.	Maximum system level clock frequencies'	878
Table 342.	Maximum system level clock frequencies	880
Table 343.	JTAG frequencies	881
Table 344.	PLL register write protection	883
Table 345.	Dual PLL Digital Interface register reset values	883
Table 346.	XOSC register write protection	885
Table 347.	XOSC register reset values	885
Table 348.	IRCOSC register reset values	885
Table 349.	Clock input sources	894
Table 350.	CMU registers availability	894
Table 351.	CMU register write protection	895
Table 352.	Dual PLL Digital Interface memory map	898
Table 353.	PLL0CR field descriptions	899
Table 354.	PLL0SR field descriptions	900
Table 355.	PLL0DV field descriptions	902
Table 356.	PLL1CR field descriptions	903
Table 357.	PLL1SR field descriptions	904
Table 358.	PLL1DV field descriptions	906
Table 359.	PLL1FM field descriptions	907
Table 360.	PLL1FD field descriptions	908
Table 361.	Signal description	915

Table 362.	CMU register summary	916
Table 363.	CMU_CSR field descriptions	917
Table 364.	CMU_FDR field descriptions	918
Table 365.	CMU_HFREFR field descriptions	919
Table 366.	CMU_LFREFR field descriptions	920
Table 367.	CMU_ISR field descriptions	920
Table 368.	CMU_MDR field descriptions	921
Table 369.	MC_CGM register description	925
Table 370.	MC_CGM memory map	930
Table 371.	CGM_AC5_CDC0 field descriptions	941
Table 372.	CGM_AC5_CDC1 field descriptions	942
Table 373.	CGM_AC5_CDC2 field descriptions	942
Table 374.	CGM_AC5_CDC10 field descriptions	943
Table 375.	CGM_AC5_CDC11 field descriptions	944
Table 376.	CGM_AC5_CDC12 field descriptions	944
Table 377.	CGM_AC5_CDC13 field descriptions	945
Table 378.	CGM_AC5_CDC20 field descriptions	946
Table 379.	CGM_AC5_CDC21 field descriptions	947
Table 380.	CGM_AC5_CDC22 field descriptions	947
Table 381.	CGM_AC5_CDC23 field descriptions	948
Table 382.	CGM_PCS_SDUR field descriptions	949
Table 383.	CGM_PCS_DIVC1 field descriptions	949
Table 384.	CGM_PCS_DIVS1 field descriptions	950
Table 385.	CGM_PCS_DIVE1 field descriptions	951
Table 386.	CGM_PCS_DIVC2 field descriptions	951
Table 387.	CGM_PCS_DIVS2 field descriptions	952
Table 388.	CGM_PCS_DIVE2 field descriptions	953
Table 389.	CGM_PCS_DIVS2 field descriptions	953
Table 390.	CGM_PCS_DIVS4 field descriptions	954
Table 391.	CGM_PCS_DIVE4 field descriptions	955
Table 392.	CGM_SC_SS field descriptions	956
Table 393.	CGM_SC_DC0 field descriptions	957
Table 394.	CGM_SC_DC1 field descriptions	958
Table 395.	CGM_SC_DC2 field descriptions	958
Table 396.	CGM_AC0_SC field descriptions	959
Table 397.	CGM_AC0_SS field descriptions	960
Table 398.	CGM_AC0_DC0 field descriptions	961
Table 399.	CGM_AC0_DC1 field descriptions	962
Table 400.	CGM_AC0_DC2 field descriptions	962
Table 401.	CGM_AC0_DC3 field descriptions	963
Table 402.	CGM_AC0_DC4 field descriptions	964
Table 403.	CGM_AC1_SC field descriptions	965
Table 404.	CGM_AC1_SS field descriptions	966
Table 405.	CGM_AC1_DC0 field descriptions	967
Table 406.	CGM_AC2_DC0 field descriptions	967
Table 407.	CGM_AC2_DC1 field descriptions	968
Table 408.	CGM_AC3_SC field descriptions	969
Table 409.	CGM_AC3_SS field descriptions	970
Table 410.	CGM_AC4_SC field descriptions	971
Table 411.	CGM_AC4_SS field descriptions	972
Table 412.	CGM_AC5_DC0 field descriptions	973
Table 413.	CGM_AC5_DC1 field descriptions	973

Table 414.	CGM_AC5_DC2 field descriptions	974
Table 415.	CGM_AC6_SC field descriptions	975
Table 416.	CGM_AC6_SS field descriptions	976
Table 417.	CGM_AC6_DC0 field descriptions	977
Table 418.	CGM_AC7_SC field descriptions	978
Table 419.	CGM_AC7_SS field descriptions	979
Table 420.	CGM_AC7_DC0 field descriptions	980
Table 421.	CGM_AC8_SC field descriptions	981
Table 422.	CGM_AC8_SS field descriptions	982
Table 423.	CGM_AC8_DC0 field descriptions	983
Table 424.	Auxiliary Clock 9 Select Control Register (CGM_AC9_SC) field descriptions	984
Table 425.	CGM_AC9_SS field descriptions	985
Table 426.	CGM_AC9_DC0 field descriptions	986
Table 427.	CGM_AC10_SC field descriptions	987
Table 428.	CGM_AC10_SS field descriptions	988
Table 429.	CGM_AC10_DC0 field descriptions	989
Table 430.	MC_CGM CGM_PCS_DIVCn[RATE] values	991
Table 431.	MC_CGM Example System Clock Division Values Compatible with Divider Synchronization	993
Table 432.	MC_CGM Example System Clock Division Values Compatible with Divider Synchronization	994
Table 433.	XOSC configurations	1008
Table 434.	XOSC memory map	1009
Table 435.	XOSC_CTL register field descriptions	1010
Table 436.	IRCOSC memory map	1012
Table 437.	CTL register field descriptions	1013
Table 438.	IRCOSC_CTL[USER_TRIM] frequency trimming calculation.	1013
Table 439.	NT register field descriptions	1014
Table 440.	TT register field descriptions.	1015
Table 441.	RAM controller memory map	1017
Table 442.	Platform RAM Configuration Register 1 description	1018
Table 443.	RAM controller basic H-matrix definition.	1021
Table 444.	RAM safety feature outputs	1024
Table 445.	Flash memory controller memory map	1028
Table 446.	PFCR1 field descriptions	1030
Table 447.	PFCR2 field descriptions	1034
Table 448.	PFCR3 field descriptions	1037
Table 449.	PFlash Access Protection Register field descriptions.	1040
Table 450.	PFCRCR field descriptions.	1044
Table 451.	PFCRDE field descriptions	1045
Table 452.	PFlash Calibration Region Descriptor <i>n</i> , Word 0 description	1052
Table 453.	PFlash Overlay Region Descriptor <i>n</i> , Word 1 description.	1053
Table 454.	PFlash Overlay Region Descriptor <i>n</i> , Word 2 description.	1054
Table 455.	Calibration data memory base addresses	1064
Table 456.	Block Memory Map.	1076
Table 457.	Test mode disable block select.	1078
Table 458.	Register conventions	1079
Table 459.	Register summary.	1080
Table 460.	MCR field descriptions	1084
Table 461.	MCRA field descriptions	1089
Table 462.	MCRE field descriptions	1091
Table 463.	LOCK0 field descriptions	1094

Table 464.	LOCK1 field descriptions	1098
Table 465.	LOCK2 field descriptions	1099
Table 466.	LOCK3 field descriptions	1100
Table 467.	LOCK0a field descriptions	1101
Table 468.	LOCK1a field descriptions	1105
Table 469.	SEL0 field descriptions	1106
Table 470.	SEL1 field descriptions	1108
Table 471.	SEL2 field descriptions	1109
Table 472.	SEL3 field descriptions	1110
Table 473.	ADR field descriptions	1111
Table 474.	UT0 field descriptions	1113
Table 475.	UM0 field descriptions	1116
Table 476.	UM1 field descriptions	1117
Table 477.	UM2 Field Descriptions	1118
Table 478.	UM3 field descriptions	1119
Table 479.	UM4 field descriptions	1120
Table 480.	UM5 field descriptions	1121
Table 481.	UM6 field descriptions	1122
Table 482.	UM7 field descriptions	1123
Table 483.	UM8 field descriptions	1124
Table 484.	UM9 field descriptions	1125
Table 485.	OPP0 field descriptions	1126
Table 486.	OPP1 field descriptions	1128
Table 487.	OPP2 field descriptions	1129
Table 488.	OPP3 field descriptions	1130
Table 489.	TMD field descriptions	1131
Table 490.	MCR bit set/clear priority levels	1131
Table 491.	Alternate program and erase characteristics	1140
Table 492.	Sample low and mid address space flash block bit mapping	1158
Table 493.	Sample low and mid address space flash block bit mapping	1170
Table 494.	Mapping of Test Mode Disable Block Select bits to flash blocks	1179
Table 495.	SPC57EM80xx analog input pin multiplexing	1187
Table 496.	Register fields for SD ADC analog input channel selection	1194
Table 497.	SDADC_0 analog input AN[x] selection	1195
Table 498.	SDADC_1 analog input AN[x] selection	1196
Table 499.	SDADC_2 analog input AN[x] selection	1197
Table 500.	SDADC_3 analog input AN[x] selection	1198
Table 501.	SDADC_4 analog input AN[x] selection	1199
Table 502.	SDADC_5 analog input AN[x] selection	1200
Table 503.	SARADC alternate references	1204
Table 504.	SAR analog input channel assignment	1208
Table 505.	SARADC_B analog test channel assignment	1208
Table 506.	SAR ADC register descriptions	1211
Table 507.	SAR ADC test channel register descriptions (SARB only)	1211
Table 508.	SAR ADC external channel register descriptions (SARB only)	1212
Table 509.	SARADC_B register definitions	1212
Table 510.	SARADC_B test channel register definitions	1218
Table 511.	SARADC_B external channel register definitions	1221
Table 512.	SARADC_0 register definitions	1225
Table 513.	SARADC_1 register definitions	1229
Table 514.	SARADC_2 register definitions	1232
Table 515.	SARADC_3 register definitions	1235

Table 516.	SARADC_4 register definitions	1238
Table 517.	SARADC_6 register definitions	1241
Table 518.	SARADC_7 register definitions	1244
Table 519.	Signal properties	1251
Table 520.	Detailed signal descriptions	1251
Table 521.	SDADC Digital interface memory map	1252
Table 522.	MCR field descriptions	1254
Table 523.	CSR field descriptions	1256
Table 524.	Analog input AN[0:7] selection	1257
Table 525.	RKR field descriptions	1258
Table 526.	SFR field descriptions	1259
Table 527.	RSER field descriptions	1261
Table 528.	OSDR field descriptions	1263
Table 529.	FCR field descriptions	1263
Table 530.	STKR field descriptions	1264
Table 531.	CDR field descriptions	1265
Table 532.	WTHHLR field descriptions	1266
Table 533.	Values of WDGxH and WDGxL fields	1282
Table 534.	SARADC digital interface register map	1284
Table 535.	MCR field descriptions	1287
Table 536.	MSR field descriptions	1290
Table 537.	ISR field descriptions	1292
Table 538.	ICIPR0–2 field descriptions	1293
Table 539.	ICIPR0–2 Registers to channel association	1293
Table 540.	IMR field descriptions	1294
Table 541.	ICIMR0–2 field descriptions	1295
Table 542.	Interrupt Mask Registers to Channel Association	1295
Table 543.	WTISR field descriptions	1296
Table 544.	WTIMR field descriptions	1297
Table 545.	DMAE register field descriptions	1298
Table 546.	ICDSR0–2 field descriptions	1299
Table 547.	Internal Channel DMA Select Registers to Channel Association	1299
Table 548.	WTHRHLR0–3 field descriptions	1300
Table 549.	CTR0–3 field descriptions	1301
Table 550.	ICNCMR0–2 field descriptions	1302
Table 551.	Internal Channel Normal Conversion Mask Registers to Channel Association	1302
Table 552.	ICJCMR0–2 field descriptions	1303
Table 553.	Internal Channel Normal Conversion Mask Registers to Channel Association	1303
Table 554.	PEDR field descriptions	1304
Table 555.	ICDR0–95 field descriptions	1305
Table 556.	WTHRHLR4–15 field descriptions	1307
Table 557.	ICWSELR0–11 field descriptions	1308
Table 558.	Internal Channel Watchdog Enable Registers to Channel Association	1308
Table 559.	ICWENR0–2 field descriptions	1309
Table 560.	Internal Channel Watchdog Enable Registers to Channel Association	1309
Table 561.	ICAWORR0–2 field descriptions	1310
Table 562.	Internal Channel Analog Watchdog Registers to Channel Association	1310
Table 563.	TCIPR field descriptions	1311
Table 564.	TCIPR Register to Channel Association	1311
Table 565.	TCIMR field descriptions	1312
Table 566.	Test Channel Interrupt Mask Register to Channel Association	1312
Table 567.	TCDSR field descriptions	1313

Table 568.	Test Channel DMA Select Registers to Channel Association.	1313
Table 569.	TCNCMR field descriptions	1314
Table 570.	Test Channel Normal Conversion Mask Register to Channel Association.	1314
Table 571.	TCJCMR field descriptions	1315
Table 572.	Test Channel Normal Conversion Mask Register to Channel Association.	1315
Table 573.	TCWSELR field descriptions	1316
Table 574.	Test Channel Watchdog Select Registers to Channel Association	1316
Table 575.	TCWENR field descriptions	1317
Table 576.	Test Channel Watchdog Enable Registers to Channel Association.	1317
Table 577.	TCAWORR field descriptions	1318
Table 578.	Analog Watchdog Register to Channel Association	1318
Table 579.	Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–7) field descriptions	1319
Table 580.	Test Channel Short Registers to Channel Association.	1319
Table 581.	TCDR96–127 field descriptions	1321
Table 582.	ECDSR field descriptions.	1322
Table 583.	ECIPR0–3 register field descriptions	1323
Table 584.	ECIPR0–3 Registers to Channel Association.	1323
Table 585.	ECIMR0–3 field descriptions	1324
Table 586.	External Channel Interrupt Mask Registers to Channel Association	1324
Table 587.	ECDSR0–3 field descriptions	1325
Table 588.	DMA Select Registers to Channel Association.	1325
Table 589.	ECNCMR0–3 field descriptions	1326
Table 590.	External Channel Normal Conversion Mask Registers to Channel Association.	1326
Table 591.	ECJCMR0–3 field descriptions	1327
Table 592.	External Channel Normal Conversion Mask Registers to Channel Association.	1327
Table 593.	ECWSELR0–15 field descriptions	1328
Table 594.	External Channel Watchdog Enable Registers to Channel Association.	1328
Table 595.	ECWENR0–3 field descriptions	1329
Table 596.	External Channel Watchdog Enable Registers to Channel Association.	1329
Table 597.	ECAWORR0–3 field descriptions	1330
Table 598.	External Channel Analog Watchdog Registers to Channel Association.	1330
Table 599.	ECMICR0–3 field descriptions	1331
Table 600.	Test Channel Short Registers to Channel Association.	1331
Table 601.	ECDR128–255 field descriptions	1332
Table 602.	Start of conversion pulse delay table	1334
Table 603.	Timers.	1341
Table 604.	STM memory map	1342
Table 605.	STM_CR field descriptions	1344
Table 606.	STM_CNT field descriptions	1344
Table 607.	STM_CCRn field descriptions.	1345
Table 608.	STM_CIRn field descriptions	1346
Table 609.	STM_CMPn field descriptions	1346
Table 610.	SWT memory map	1348
Table 611.	SWT_CR field descriptions.	1349
Table 612.	SWT_IR field descriptions	1350
Table 613.	SWT_TO field descriptions	1351
Table 614.	SWT_WN field descriptions	1351
Table 615.	SWT_SR field descriptions.	1352
Table 616.	SWT_CO field descriptions.	1352
Table 617.	SWT_SK field descriptions	1353
Table 618.	PITx memory map	1357

Table 619.	PITx_MCR field descriptions	1359
Table 620.	PITx_LTMR64H field descriptions	1360
Table 621.	PITx_LTMR64L field descriptions	1361
Table 622.	PITx_LDVALn field descriptions	1362
Table 623.	PITx_CVALn field descriptions	1362
Table 624.	PITx_TCTRLn field descriptions	1363
Table 625.	PITx_TFLGn field descriptions	1364
Table 626.	PITx_LDVALn field descriptions	1365
Table 627.	PITx_CVALn field descriptions	1365
Table 628.	PITx_TCTRLn field descriptions	1366
Table 629.	PITx_TFLGn field descriptions	1367
Table 630.	References	1372
Table 631.	M_CAN memory map	1378
Table 632.	Core Release Register Field Descriptions	1380
Table 633.	Endian Register Field Descriptions	1381
Table 634.	Test Register field descriptions	1383
Table 635.	RAM Watchdog register field descriptions	1384
Table 636.	CC Control Register field descriptions	1385
Table 637.	Bit Timing and Prescaler Field Descriptions	1388
Table 638.	Timestamp Counter Configuration field descriptions	1389
Table 639.	Timestamp counter value field descriptions	1390
Table 640.	Timeout Counter Configuration Field Descriptions	1391
Table 641.	Timeout counter value field descriptions	1392
Table 642.	Error counter register field descriptions	1393
Table 643.	PSR descriptions	1394
Table 644.	Interrupt register field descriptions	1396
Table 645.	Interrupt enable register field descriptions	1399
Table 646.	Interrupt line select register field descriptions	1402
Table 647.	Interrupt line enable register field descriptions	1405
Table 648.	Global filter configuration register field descriptions	1406
Table 649.	Standard ID Filter Configuration register field descriptions	1407
Table 650.	Extended ID Filter Configuration register field descriptions	1408
Table 651.	Extended ID and Mask register field descriptions	1409
Table 652.	High Priority Message Status register field descriptions	1410
Table 653.	Rx FIFO 0 Configuration register field descriptions	1412
Table 654.	Rx FIFO 0 Status Register	1413
Table 655.	Rx FIFO 0 Status register field descriptions	1413
Table 656.	Rx FIFO 0 Acknowledge register field descriptions	1414
Table 657.	Rx FIFO 0 Configuration register field descriptions	1415
Table 658.	Rx FIFO 0 Configuration register field descriptions	1416
Table 659.	Rx FIFO 1 Status register field descriptions	1416
Table 660.	Rx FIFO 1 Acknowledge register field descriptions	1418
Table 661.	Tx Buffer Configuration register field descriptions	1418
Table 662.	Tx Buffer Configuration register field descriptions	1419
Table 663.	Tx Buffer Request Pending register	1420
Table 664.	Tx Buffer Request Pending register field descriptions	1420
Table 665.	Tx Buffer Add Request register	1421
Table 666.	Tx Buffer Add Request register field descriptions	1421
Table 667.	Tx Buffer Cancellation Request register (TXBCR)	1421
Table 668.	TXBCR field descriptions	1422
Table 669.	Tx Buffer Transmission Occurred field description	1422
Table 670.	Tx Buffer Cancellation Finished field description	1423

Table 671.	Tx Buffer Transmission Interrupt field description	1424
Table 672.	Tx Buffer Cancellation Finished Interrupt Enable field description	1424
Table 673.	Tx Event FIFO Configuration field description	1425
Table 674.	Tx Event FIFO Status field description	1426
Table 675.	Tx Event FIFO Acknowledge field description	1427
Table 676.	Rx FIFO Element descriptions	1429
Table 677.	Tx Buffer Element descriptions	1430
Table 678.	Tx Event FIFO Element	1431
Table 679.	Standard Message ID Filter Element Field Description	1433
Table 680.	Extended Message ID Filter Element Field Description	1434
Table 681.	Coding of DLC in CAN FD	1439
Table 682.	Example Filter Configuration for Rx buffers	1449
Table 683.	Example Filter Configuration for debug messages	1450
Table 684.	M_TTCAN Memory Map	1459
Table 685.	Core Release register field descriptions	1462
Table 686.	Endian Register Field Descriptions	1463
Table 687.	Fast Bit Timing and Prescaler Register	1464
Table 688.	Test Register field descriptions	1465
Table 689.	RAM Watchdog register field descriptions	1467
Table 690.	CC Control Register Field Descriptions	1467
Table 691.	Bit Timing and Prescaler Field Descriptions	1470
Table 692.	Timestamp Counter Configuration Field Descriptions	1471
Table 693.	Timestamp counter value field descriptions	1472
Table 694.	Timeout counter configuration field descriptions	1473
Table 695.	Timeout counter value field descriptions	1474
Table 696.	Error counter register field descriptions	1475
Table 697.	PSR descriptions	1475
Table 698.	Interrupt register field descriptions	1478
Table 699.	Interrupt enable register field descriptions	1481
Table 700.	Interrupt line select register field descriptions	1484
Table 701.	Interrupt line enable register field descriptions	1487
Table 702.	Global filter configuration register field descriptions	1488
Table 703.	Standard ID Filter Configuration register field descriptions	1489
Table 704.	Extended ID Filter Configuration register field descriptions	1490
Table 705.	Extended ID and Mask register field descriptions	1491
Table 706.	High Priority Message Status register field descriptions	1492
Table 707.	Rx FIFO 0 Configuration register field descriptions	1494
Table 708.	Rx FIFO 0 Status register field descriptions	1495
Table 709.	Rx FIFO 0 Acknowledge register field descriptions	1496
Table 710.	Rx FIFO 0 Configuration register field descriptions	1497
Table 711.	Rx FIFO 1 Configuration register field descriptions	1497
Table 712.	Rx FIFO 1 Status register field descriptions	1498
Table 713.	Rx FIFO 1 Acknowledge register field description	1499
Table 714.	Tx Buffer Configuration register field description	1500
Table 715.	Tx FIFO/Queue Status register field description	1501
Table 716.	Tx Buffer Request Pending register field description	1502
Table 717.	Tx Buffer and Request register field description	1503
Table 718.	Tx Buffer Cancellation Request register field description	1503
Table 719.	Tx Buffer Transmission Occurred register field description	1504
Table 720.	Tx Buffer Cancellation Finished register field description	1505
Table 721.	Tx Buffer Transmission Interrupt Enable register field description	1505
Table 722.	Tx Buffer Cancellation Finished Interrupt Enable register field description	1506

Table 723.	Tx Event FIFO Configuration register field description	1507
Table 724.	Tx Event FIFO Status register field description	1507
Table 725.	Tx Event FIFO Acknowledge register field description	1508
Table 726.	TT Trigger Memory Configuration register field description	1509
Table 727.	TT Reference Message Configuration register field description	1510
Table 728.	TT operation Configuration register field description	1511
Table 729.	TT Matrix Limits register field description	1512
Table 730.	TUR Configuration register field description	1514
Table 731.	TT Operation Control register field description	1515
Table 732.	TT Global Time Preset register field description	1517
Table 733.	TT Time Mark register field description	1518
Table 734.	TT Interrupt Register field description	1519
Table 735.	TT Interrupt Enable register field description	1521
Table 736.	TT Interrupt Line Select register field description	1523
Table 737.	TT Operation Status register field description	1525
Table 738.	TUR Numerator Actual register field description	1527
Table 739.	TT Local and Global Time register field description	1528
Table 740.	TT Cycle Time and Count register field description	1529
Table 741.	TT Capture Time register field description	1529
Table 742.	TT Cycle Sync Mark register field description	1530
Table 743.	Rx Buffer and FIFO Element descriptions	1532
Table 744.	Tx Buffer Element descriptions	1534
Table 745.	Tx Event FIFO Element	1535
Table 746.	Standard Message ID Filter Element Field Description	1536
Table 747.	Extended Message ID Filter Element Field Description	1537
Table 748.	Trigger Memory Element Field Description	1538
Table 749.	Coding of DLC in CAN FD	1541
Table 750.	Example Filter Configuration for Rx Buffers	1551
Table 751.	System Matrix Node A	1562
Table 752.	Trigger List Node A	1563
Table 753.	Number of Data Bytes transmitted with a Reference Messages	1570
Table 754.	M_CAN message memory size	1578
Table 755.	M_TTCAN message memory size	1579
Table 756.	Shared memory map vs CAN transmit, receive and filters elements	1583
Table 757.	Detailed mapping in CAN subblocks	1584
Table 758.	SIPI header command coding	1590
Table 759.	SIPI Header Channel Number Coding	1591
Table 760.	Payload size of LFAST channel frame	1592
Table 761.	Converted coding of LFAST channel code for SIPI headers	1592
Table 762.	SIPI memory map	1610
Table 763.	CCRn field descriptions	1613
Table 764.	CSRn field descriptions	1614
Table 765.	CIRn field descriptions	1615
Table 766.	CTORn field descriptions	1616
Table 767.	CCRCn field descriptions	1617
Table 768.	CARn field descriptions	1618
Table 769.	CDRn field descriptions	1619
Table 770.	Channel data Register address offsets	1619
Table 771.	SIPIMCR field descriptions	1620
Table 772.	SIPIISR field descriptions	1621
Table 773.	MAXCR field descriptions	1623
Table 774.	SIPIARR field descriptions	1623

Table 775.	SIPIACR field descriptions	1624
Table 776.	SIPIERR field descriptions	1625
Table 777.	DSPI signal description	1632
Table 778.	DSPI memory map	1633
Table 779.	DSPIx_MCR field descriptions	1637
Table 780.	DSPIx_TCR field descriptions	1640
Table 781.	DSPIx_CTAR _n field descriptions	1642
Table 782.	DSPI SCK duty cycle	1644
Table 783.	Delay scaler encoding	1645
Table 784.	DSPI baud rate scaler	1645
Table 785.	DSPIx_CTAR _n _SLAVE field descriptions	1646
Table 786.	DSPIx_SR field descriptions	1648
Table 787.	DSPIx_RSER field descriptions	1651
Table 788.	DSPIx_PUSHR field descriptions	1655
Table 789.	DSPIx_PUSHR_SLAVE field descriptions	1657
Table 790.	DSPIx_POPR field descriptions	1658
Table 791.	DSPIx_TXFR _n field descriptions	1659
Table 792.	DSPIx_RXFR _n field descriptions	1660
Table 793.	DSPIx_DSICR0 field descriptions	1661
Table 794.	DSPIx_SDR0 field descriptions	1664
Table 795.	DSPIx_ASDR0 field descriptions	1665
Table 796.	DSPIx_COMPR0 field descriptions	1666
Table 797.	DSPIx_DDR0 field descriptions	1667
Table 798.	DSPIx_DSICR1 field descriptions	1668
Table 799.	DSPIx_SSR0 field descriptions	1670
Table 800.	DSPIx_PISR0 field descriptions	1671
Table 801.	DSPIx_PISR1 field descriptions	1673
Table 802.	DSPIx_PISR2 field descriptions	1674
Table 803.	DSPIx_PISR3 field descriptions	1675
Table 804.	DSPIx_DIMR0 field descriptions	1676
Table 805.	DSPIx_DPIR0 field descriptions	1677
Table 806.	DSPIx_SDR1 field descriptions	1678
Table 807.	DSPIx_ASDR1 field descriptions	1679
Table 808.	DSPIx_COMPR1 field descriptions	1680
Table 809.	DSPIx_DDR1 field descriptions	1681
Table 810.	DSPIx_SSR field descriptions	1682
Table 811.	DSPIx_PISR4 field descriptions	1683
Table 812.	DSPIx_PISR5 field descriptions	1685
Table 813.	DSPIx_PISR6 field descriptions	1686
Table 814.	DSPIx_PISR7 field descriptions	1687
Table 815.	DSPIx_DIMR1 field descriptions	1688
Table 816.	DSPIx_DPIR1 field descriptions	1689
Table 817.	DSPIx_CTARE _n field descriptions	1690
Table 818.	DSPIx_SREX field descriptions	1691
Table 819.	DSI data transfer initiation control	1701
Table 820.	Baud rate computation example	1705
Table 821.	PCS to SCK delay computation example	1705
Table 822.	After SCK delay computation example	1706
Table 823.	Delay after transfer computation example	1706
Table 824.	Peripheral Chip Select Strobe Assert Computation Example	1707
Table 825.	Peripheral Chip Select Strobe Negate Computation Example	1707
Table 826.	Differences between TSB and ITSB modes	1723

Table 827.	Interrupt and DMA request conditions	1726
Table 828.	Baud rate values (bps)	1731
Table 829.	Delay values	1732
Table 830.	Operating modes for LFAST interface	1736
Table 831.	Header payload sizes	1738
Table 832.	LFAST memory map	1739
Table 833.	Register conventions	1741
Table 834.	MCR field descriptions	1742
Table 835.	SCR field descriptions	1744
Table 836.	COCR field descriptions	1746
Table 837.	TMCR field descriptions	1747
Table 838.	ALCR field descriptions	1749
Table 839.	RCDCR field description	1750
Table 840.	SLCR field descriptions	1751
Table 841.	ICR field descriptions	1752
Table 842.	PICR field descriptions	1753
Table 843.	RFCR field descriptions	1754
Table 844.	TIER field descriptions	1755
Table 845.	RIER field descriptions	1756
Table 846.	RIIER field descriptions	1757
Table 847.	PLLCR field descriptions	1759
Table 848.	LCR field description	1761
Table 849.	UNSTCR field descriptions	1763
Table 850.	UNSTDR _n field descriptions	1764
Table 851.	GSR field descriptions	1765
Table 852.	PISR field descriptions	1766
Table 853.	DFSR field descriptions	1767
Table 854.	TISR field descriptions	1767
Table 855.	RISR field descriptions	1769
Table 856.	RIISR field descriptions	1770
Table 857.	PLLLSR field descriptions	1772
Table 858.	UNSRSR field descriptions	1773
Table 859.	UNSRDR _n field descriptions	1773
Table 860.	High speed 8 phase selection - sampling procedure	1780
Table 861.	High speed 4 phase selection - sampling procedure	1781
Table 862.	Low speed 4 phase selection - sampling procedure	1781
Table 863.	Line Receiver State	1785
Table 864.	Priority Levels for the Transmitter Controller	1786
Table 865.	Line Driver States	1789
Table 866.	Frames supported by LFAST interfaces	1790
Table 867.	Supported ICLC Payloads	1793
Table 868.	Loopback modes	1796
Table 869.	Recommended transmit exception handling	1803
Table 870.	Recommended receive exception handling mechanism	1804
Table 871.	Frames Supported by LFAST interfaces	1805
Table 872.	Recommended receive exception handling mechanism	1806
Table 873.	Rx clocks summary	1811
Table 874.	Tx clocks summary	1811
Table 875.	FEC sub-block descriptions	1815
Table 876.	FEC signal descriptions	1818
Table 877.	FEC memory map	1819
Table 878.	FEC register map	1820

Table 879.	EIR field descriptions	1823
Table 880.	EIMR field descriptions	1824
Table 881.	RDAR field descriptions	1826
Table 882.	TDAR field descriptions	1827
Table 883.	ECR field descriptions	1827
Table 884.	MMFR field descriptions	1828
Table 885.	MSCR field descriptions	1829
Table 886.	Programming examples for MSCR	1830
Table 887.	MIBC field descriptions	1830
Table 888.	RCR field descriptions	1831
Table 889.	TCR field descriptions	1833
Table 890.	PALR field descriptions	1834
Table 891.	PAUR field descriptions	1834
Table 892.	OPD field descriptions	1835
Table 893.	IAUR field descriptions	1835
Table 894.	IALR field descriptions	1836
Table 895.	GAUR field descriptions	1836
Table 896.	GALR field descriptions	1837
Table 897.	TFWR field descriptions	1837
Table 898.	FRBR field descriptions	1838
Table 899.	FRSR field descriptions	1838
Table 900.	ERDSR field descriptions	1839
Table 901.	ETDSR field descriptions	1839
Table 902.	EMRBR field descriptions	1840
Table 903.	RMON_T_DROP field descriptions	1840
Table 904.	RMON_T_PACKETS field descriptions	1841
Table 905.	RMON_T_PACKETS field descriptions	1841
Table 906.	RMON_T_MC_PKT field descriptions	1841
Table 907.	RMON_T_CRC_ALIGN field descriptions	1842
Table 908.	RMON_T_CRC_ALIGN field descriptions	1842
Table 909.	RMON_T_OVERSIZE field descriptions	1842
Table 910.	RMON_T_FRAG field descriptions	1843
Table 911.	RMON_T_JAB field descriptions	1843
Table 912.	RMON_T_COL field descriptions	1843
Table 913.	RMON_T_P64 field descriptions	1844
Table 914.	RMON_T_P65TO127 field descriptions	1844
Table 915.	RMON_T_P128TO255 field descriptions	1844
Table 916.	RMON_T_P256TO511 field descriptions	1845
Table 917.	RMON_T_P512TO1023 field descriptions	1845
Table 918.	RMON_T_P1024TO2047 field descriptions	1845
Table 919.	RMON_T_P_GTE2048 field descriptions	1846
Table 920.	RMON_T_OCTETS field descriptions	1846
Table 921.	IEEE_T_DROP field descriptions	1846
Table 922.	IEEE_T_FRAME_OK field descriptions	1847
Table 923.	IEEE_T_1COL field descriptions	1847
Table 924.	IEEE_T_MCOL field descriptions	1847
Table 925.	IEEE_T_DEF field descriptions	1848
Table 926.	IEEE_T_LCOL field descriptions	1848
Table 927.	IEEE_T_EXCOL field descriptions	1848
Table 928.	IEEE_T_MACERR field descriptions	1849
Table 929.	IEEE_T_CSERR field descriptions	1849
Table 930.	IEEE_T_SQE field descriptions	1849

Table 931.	IEEE_T_FDXFC field descriptions	1850
Table 932.	IEEE_T_OCTETS_OK field descriptions	1850
Table 933.	RMON_R_DROP field descriptions	1851
Table 934.	RMON_R_PACKETS field descriptions	1851
Table 935.	RMON_R_BC_PKT field descriptions	1851
Table 936.	RMON_R_MC_PKT field descriptions	1852
Table 937.	RMON_R_CRC_ALIGN field descriptions	1852
Table 938.	RMON_R_UNDERSIZE field descriptions	1852
Table 939.	RMON_R_OVERSIZE field descriptions	1853
Table 940.	RMON_R_FRAG field descriptions	1853
Table 941.	RMON_R_JAB field descriptions	1853
Table 942.	RMON_R_RESVD_0 field descriptions	1854
Table 943.	RMON_R_P64 field descriptions	1854
Table 944.	RMON_R_P65TO127 field descriptions	1854
Table 945.	RMON_R_P128TO255 field descriptions	1855
Table 946.	RMON_R_P256TO511 field descriptions	1855
Table 947.	RMON_R_P512TO1023 field descriptions	1855
Table 948.	RMON_R_P1024TO2047 field descriptions	1856
Table 949.	RMON_R_P_GTE2048 field descriptions	1856
Table 950.	RMON_R_OCTETS field descriptions	1856
Table 951.	IEEE_R_DROP field descriptions	1857
Table 952.	IEEE_R_FRAME_OK field descriptions	1857
Table 953.	IEEE_R_CRC field descriptions	1857
Table 954.	IEEE_R_ALIGN field descriptions	1858
Table 955.	IEEE_R_MACERR field descriptions	1858
Table 956.	IEEE_R_FDXFC field descriptions	1858
Table 957.	IEEE_R_OCTETS_OK field descriptions	1859
Table 958.	MII data frame components	1859
Table 959.	MII management frame components	1860
Table 960.	MII management register set	1861
Table 961.	Receive Buffer Descriptor field descriptions	1863
Table 962.	Transmit Buffer Descriptor field descriptions	1865
Table 963.	ECR[ETHER_EN] De-Assertion Effect on FEC	1866
Table 964.	MII mode	1868
Table 965.	RMII mode configuration	1868
Table 966.	7-Wire mode configuration	1868
Table 967.	Destination address to 6-bit hash	1874
Table 968.	PAUSE frame field specification	1876
Table 969.	List of terms	1880
Table 970.	External Signal Properties	1886
Table 971.	FlexRay memory map	1889
Table 972.	Register access conventions	1895
Table 973.	Additional register reset conditions	1895
Table 974.	Register Write Access Restrictions	1896
Table 975.	FR_MVR field descriptions	1897
Table 976.	FR_MCR field descriptions	1898
Table 977.	FlexRay channel selection	1899
Table 978.	FR_SYMBADR field descriptions	1900
Table 979.	FR_STBSCR field descriptions	1901
Table 980.	Strobe signal mapping	1901
Table 981.	FR_MBDSR field descriptions	1902
Table 982.	FR_MBSSUTR Field Descriptions	1903

Table 983.	FR_PEDRAR field descriptions	1904
Table 984.	FR_POCR Field Descriptions	1905
Table 985.	FR_GIFER field descriptions	1907
Table 986.	FR_PIFR0 field descriptions	1909
Table 987.	FR_PIFR1 field descriptions	1912
Table 988.	FR_PIER0 field descriptions	1913
Table 989.	FR_PIER1 field descriptions	1914
Table 990.	FR_CHIERFR field descriptions	1916
Table 991.	FR_MBIVEC field descriptions	1918
Table 992.	FR_CASERCR Field Descriptions	1919
Table 993.	FR_CBSERCR field descriptions	1919
Table 994.	FR_PSR0 field descriptions	1920
Table 995.	FR_PSR1 field descriptions	1921
Table 996.	FR_PSR2 field descriptions	1922
Table 997.	FR_PSR3 field descriptions	1924
Table 998.	FR_MTCTR field descriptions	1925
Table 999.	FR_CYCTR field descriptions	1926
Table 1000.	FR_SLTCTAR field descriptions	1926
Table 1001.	FR_SLTCTBR field descriptions	1927
Table 1002.	FR_RTCORVR Field Descriptions	1927
Table 1003.	FR_OFCORVR Field Descriptions	1928
Table 1004.	FR_CIFR field descriptions	1929
Table 1005.	FR_SYMATOR field descriptions	1930
Table 1006.	FR_SFCNTR Field Descriptions	1930
Table 1007.	FR_SFTOR field description	1931
Table 1008.	FR_SFTCCSR field descriptions	1931
Table 1009.	FR_SFIDRFR field descriptions	1933
Table 1010.	FR_SFIDAFVR Field Descriptions	1933
Table 1011.	FR_SFIDAFMR Field Descriptions	1934
Table 1012.	NMVR[0:5] field descriptions	1935
Table 1013.	Mapping of NMVRn to the Received Payload Bytes NMVn	1935
Table 1014.	FR_NMVLr field descriptions	1935
Table 1015.	FR_TICCR field descriptions	1936
Table 1016.	FR_TI1CYSR Field Descriptions	1937
Table 1017.	FR_TI1MTOR Field Descriptions	1937
Table 1018.	FR_TI2CR0 field descriptions	1938
Table 1019.	FR_TI2CR1 field descriptions	1939
Table 1020.	FR_SSSR field descriptions	1939
Table 1021.	Mapping between FR_SSSRn and FR_SSRn	1940
Table 1022.	FR_SSCCR field descriptions	1941
Table 1023.	Mapping between internal FR_SSCCRn and FR_SSCRn	1942
Table 1024.	FR_SSR0–FR_SSR7 field descriptions	1942
Table 1025.	FR_SSCR0–FR_SSCR3 field descriptions	1944
Table 1026.	FR_MTSACFR field descriptions	1945
Table 1027.	FR_MTSBCFR field descriptions	1945
Table 1028.	FR_RSBR field descriptions	1946
Table 1029.	FR_RFSR field descriptions	1946
Table 1030.	FR_RFSYMBADR field descriptions	1947
Table 1031.	FR_RFPTR field descriptions	1948
Table 1032.	SEL-controlled receiver FIFO registers	1948
Table 1033.	FR_RFWMSR field descriptions	1948
Table 1034.	FR_RFSIR field descriptions	1949

Table 1035. RFDSR Field Descriptions	1949
Table 1036. FR_RFARIR field descriptions	1950
Table 1037. FR_RFBIR field descriptions	1950
Table 1038. FR_RFFLPCR field descriptions	1951
Table 1039. FR_RFMDAFVR field descriptions	1951
Table 1040. FR_RFMDAFMR field descriptions	1952
Table 1041. FR_RFFIDRFVR field descriptions	1952
Table 1042. FR_RFFIDRFMR field descriptions	1953
Table 1043. FR_RFRFCFR field descriptions	1953
Table 1044. FR_RFRFCTR field descriptions	1954
Table 1045. FR_LDTXSLAR field descriptions	1955
Table 1046. FR_LDTXSLBR field descriptions	1956
Table 1047. Protocol Configuration Register fields	1956
Table 1048. Wakeup channel selection	1958
Table 1049. FR_EEIFER field descriptions	1967
Table 1050. FR_EERICR Field Descriptions	1969
Table 1051. FR_EERAR Field Descriptions	1970
Table 1052. LRAM bank value for MID = 1	1970
Table 1053. FR_EERDR field descriptions	1971
Table 1054. Valid bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field	1971
Table 1055. FR_EERSR field descriptions	1972
Table 1056. FR_EEIAR field descriptions	1973
Table 1057. FR_EEIDR field descriptions	1973
Table 1058. FR_EEICR field descriptions	1974
Table 1059. FR_MBCCSRn field descriptions	1975
Table 1060. FR_MBCCFRn field descriptions	1976
Table 1061. Channel assignment description	1976
Table 1062. FR_MBFIDRn field descriptions	1977
Table 1063. FR_MBDXRn field descriptions	1978
Table 1064. FR_MBDORn field descriptions	1978
Table 1065. FR_LEETRn field descriptions	1978
Table 1066. FR_PEOER field descriptions	1979
Table 1067. FR_STPWR field descriptions	1980
Table 1068. Frame header write access constraints (transmit message buffer)	1993
Table 1069. Frame header field descriptions (receive message buffer and receive FFO)	1994
Table 1070. Frame header field descriptions (transmit message buffer)	1994
Table 1071. Receive message buffer slot status content	1995
Table 1072. Receive message buffer slot status field description	1996
Table 1073. Transmit message buffer slot status content	1997
Table 1074. Transmit message buffer slot status structure field description	1998
Table 1075. Message buffer data field minimum length	1999
Table 1076. Frame data write access constraints	2000
Table 1077. Frame data field descriptions	2000
Table 1078. Individual message buffer types	2002
Table 1079. Transmit message buffer access regions description	2003
Table 1080. Transmit message buffer state description	2004
Table 1081. Transmit message buffer application transitions	2005
Table 1082. Transmit message buffer module transitions	2006
Table 1083. Transmit message buffer transition priorities	2007
Table 1084. Receive message buffer access region description	2011
Table 1085. Receive message buffer states and access	2012
Table 1086. Receive message buffer application transitions	2013

Table 1087. Receive message buffer module transitions	2014
Table 1088. Receive message buffer transition priorities	2014
Table 1089. Receive message buffer update	2015
Table 1090. Message buffer search priority (static segment)	2018
Table 1091. Message buffer search priority (dynamic segment)	2018
Table 1092. Sync frame table generation modes	2031
Table 1093. Key slot frame type	2034
Table 1094. Slot status content	2038
Table 1095. FlexRay channel bit rate control	2047
Table 1096. PE DRAM layout	2047
Table 1097. CHI LRAM layout	2049
Table 1098. Detected memory error types	2050
Table 1099. PE DRAM checkbits coding	2051
Table 1100. FR_EECCR[CODE] PE DRAM syndrome coding	2051
Table 1101. CHI LRAM checkbits coding	2052
Table 1102. FR_EECCR[CODE] CHI LRAM syndrome coding	2052
Table 1103. Maximum SYMATOR[TIMEOUT] examples	2057
Table 1104. Minimum f_{chi} [MHz] examples (128 message buffers used)	2061
Table 1105. Protocol control command priorities	2061
Table 1106. Transmit buffer configuration	2062
Table 1107. Receive buffer configuration	2063
Table 1108. Chip modes supported by the I ² C module	2067
Table 1109. Module-specific modes supported by the I ² C module	2068
Table 1110. I ² C Conditions	2068
Table 1111. External signal descriptions	2069
Table 1112. I ² C memory map	2070
Table 1113. IBAD field descriptions	2071
Table 1114. IBFD field descriptions	2071
Table 1115. IBCR field descriptions	2072
Table 1116. IBSR field descriptions	2074
Table 1117. IBDR field descriptions	2076
Table 1118. IBIC field descriptions	2076
Table 1119. IBDBG field descriptions	2077
Table 1120. I ² C definitions	2078
Table 1121. I ² C high-level protocol steps	2079
Table 1122. Slave address transmission process	2079
Table 1123. I-Bus multiplier factor	2082
Table 1124. I-Bus prescaler divider values	2082
Table 1125. I-Bus tap and prescale values	2082
Table 1126. I ² C divider and hold values	2085
Table 1127. Interrupt summary	2090
Table 1128. Normal operating states	2108
Table 1129. State transition table	2111
Table 1130. PSI5 port list	2112
Table 1131. PSI5 memory map	2112
Table 1132. PSI5_GCR field descriptions	2137
Table 1133. PSI5_PCCR field descriptions	2138
Table 1134. PSI5_DCR field descriptions	2144
Table 1135. PSI5_DSR field descriptions	2146
Table 1136. PSI5_GICR field descriptions	2148
Table 1137. PSI5_NDICR field descriptions	2150
Table 1138. PSI5_OWICR field descriptions	2150

Table 1139. PSI5_EICR field descriptions	2151
Table 1140. PSI5_GISR field descriptions	2152
Table 1141. PSI5_DPMR field descriptions	2154
Table 1142. PSI5_DSFR field descriptions	2155
Table 1143. PSI5_DDSD field descriptions	2156
Table 1144. PSI5_PMRRL field description	2156
Table 1145. PSI5_PMRRLH field descriptions	2157
Table 1146. PSI5_PMRRLn field descriptions	2159
Table 1147. PSI5_PMRRLHn field descriptions	2159
Table 1148. PSI5_SFRn field descriptions	2161
Table 1149. PSI5_NDSR field descriptions	2163
Table 1150. OWSR field descriptions	2164
Table 1151. PSI5_EISR field descriptions	2164
Table 1152. PSI5_SNDSR field descriptions	2165
Table 1153. PSI5_SOWSR field descriptions	2166
Table 1154. PSI5_SEISR field descriptions	2166
Table 1155. PSI5_SSESR field description	2167
Table 1156. PSI5_STSR field descriptions	2168
Table 1157. DTSRR field descriptions	2169
Table 1158. PSI5_SnFCR field descriptions	2169
Table 1159. PSI5_SnSBR field descriptions	2170
Table 1160. PSI5_SnEBR field descriptions	2172
Table 1161. PSI5_DOBCR field descriptions	2173
Table 1162. DOBCR bit configurations and the related output states	2175
Table 1163. PSI5_MDDIS_OFF field descriptions	2176
Table 1164. PSI5_PW0D field descriptions	2176
Table 1165. PSI5_PW1D field descriptions	2177
Table 1166. PSI5_CTPR field descriptions	2178
Table 1167. PSI5_CIPR field descriptions	2179
Table 1168. DPRL PSI5_ field descriptions	2180
Table 1169. PSI5_DPRH field descriptions	2181
Table 1170. PSI5_DBRL field descriptions	2182
Table 1171. PSI5_DBRH field descriptions	2186
Table 1172. PSI5_DSRL field descriptions	2187
Table 1173. PSI5_DSRH field descriptions	2188
Table 1174. CRC Calculation (at transmitter side)	2211
Table 1175. Transmitted and Received Sensor Data	2212
Table 1176. CRC Re-Calculation (At Receiver Side)	2212
Table 1177. State transition Table	2242
Table 1179. PSI5-S memory map	2245
Table 1180. PS_LINCR1 field descriptions	2256
Table 1181. PS_LINIER field descriptions	2257
Table 1182. LINSR field description	2259
Table 1183. PS_UARTCR field descriptions	2260
Table 1184. PS_UARTSR field descriptions	2262
Table 1185. PS_LINFBRR field descriptions	2265
Table 1186. PS_LINIBRR field descriptions	2266
Table 1187. PS_LINCR2 field descriptions	2267
Table 1188. PS_BDRL field descriptions	2268
Table 1189. PS_BDRM field descriptions	2269
Table 1190. PS_GCR field descriptions	2270
Table 1191. PS_UARTPTO field descriptions	2271

Table 1192. PS_UARTCTO field descriptions	2272
Table 1193. DMATXE field descriptions	2273
Table 1194. DMARXE field descriptions	2274
Table 1195. PS_GLCR bit field details	2275
Table 1196. PS_GLSR bit field details	2278
Table 1197. PS_CH_BASE_ADDR bit field details	2278
Table 1198. PS_MRU_BUF2_REG0 bit field details	2279
Table 1199. PS_MRU_BUF2_REG1 bit field details	2280
Table 1200. PS_MRU_BUF2_REG2 bit field details	2282
Table 1201. PS_MRU_BUF2_REG3 bit field details	2283
Table 1202. PS_WD_DDTRIG_CLKDIV0 bit field details	2283
Table 1203. PS_WD_DDTRIG_CLKDIV1 bit field details	2284
Table 1204. PS_WD_DDTRIG_CLKDIV2 bit field details	2285
Table 1205. PS_WD_DDTRIG_CLKDIV2 bit field details	2286
Table 1206. PS_MBOX_SR_IRQ bit field details	2287
Table 1207. PS_ERR_SR_IRQ bit field details	2289
Table 1208. PS_MBOX_SEL_irq[irq_n] bit field details	2293
Table 1209. PS_ERR_SEL_IRQ[irq_n] bit field details	2295
Table 1210. PS_WDGTSSR bit field details	2297
Table 1211. PS_MSGA_CH0 bit field details	2298
Table 1212. PS_MSGB_CH0 bit field details	2299
Table 1213. PS_MBOX_SR_CH0 bit field details	2299
Table 1214. PS_MSGA_CHn bit field details	2301
Table 1215. PS_MSGB_CHn bit field details	2304
Table 1216. PS_MBOX_SR_CHn bit field details	2306
Table 1217. PS_WD_CFGR_CHn bit field details	2310
Table 1218. PS_DDTRIG_OFFR_CHn	2312
Table 1219. PS_DDTRIG_PERR_CHn	2312
Table 1220. PS_E2SCR_CHn bit details	2313
Table 1221. PS_E2SSR_CHn bit field details	2316
Table 1222. PS_DDSR_H_CHn bit details	2317
Table 1223. PS_DDSR_L_CHn bit field details	2318
Table 1224. CRC Calculation (at transmitter side)	2326
Table 1225. Transmitted and Received Sensor Data	2327
Table 1226. CRC Re-Calculation (At Receiver Side)	2327
Table 1227. SENT memory map	2348
Table 1228. GBL_CTRL field descriptions	2350
Table 1229. CHNL_EN field descriptions	2352
Table 1230. GBL_STATUS field descriptions	2353
Table 1231. FMSG_RDY field descriptions	2356
Table 1232. SMSG_RDY field descriptions	2358
Table 1233. DATA_CTRL1 field descriptions	2359
Table 1234. DATA_CTRL2 field descriptions	2361
Table 1235. FDMA_CTRL field descriptions	2363
Table 1236. SDMA_CTRL field descriptions	2364
Table 1237. FRDY_IE field descriptions	2365
Table 1238. SRDY_IE field descriptions	2366
Table 1239. DMA_FMSG_DATA field descriptions	2367
Table 1240. DMA_FMSG_CRC field descriptions	2369
Table 1241. DMA_FMSG_TS field descriptions	2370
Table 1242. DMA_SMSG_BIT3 Field Descriptions	2371
Table 1243. DMA_SMSG_BIT2 field descriptions	2372

Table 1244. DMA_SMSG_TS field descriptions	2373
Table 1245. CHn_CLK_CTRL field descriptions	2374
Table 1246. CHn_STATUS field descriptions	2377
Table 1247. CHn_CONFIG field descriptions	2380
Table 1248. 10-bit-shift-register sample register	2414
Table 1249. BDRL access in UART mode	2422
Table 1250. BDRM access in UART mode	2423
Table 1251. LINFlexD control/status fields description	2425
Table 1252. LINFlexD DMA control fields description	2427
Table 1253. TCD control fields description	2427
Table 1254. Master node — Tx mode — Register setting	2429
Table 1255. TCD setting — Master node — Tx mode	2429
Table 1256. TCD setting — Master node — Rx mode	2431
Table 1257. Slave node — Tx mode — Register setting	2433
Table 1258. TCD setting — Slave node — Tx mode	2433
Table 1259. Slave node — Rx mode — Register setting	2435
Table 1260. TCD setting — Slave node — Rx mode	2436
Table 1261. TCD setting — UART — Tx mode	2438
Table 1262. TCD setting — UART — Rx mode	2440
Table 1263. LINFlexD memory map	2443
Table 1264. LINC1R1 field descriptions	2445
Table 1265. LINIER field descriptions	2448
Table 1266. LINSR field description	2450
Table 1267. LINESR field descriptions	2453
Table 1268. UARTCR field descriptions	2456
Table 1269. UARTSR field descriptions	2460
Table 1270. LINTCSR field descriptions	2463
Table 1271. LINOCR field descriptions	2464
Table 1272. LINTOCR field descriptions	2465
Table 1273. LINFBR field descriptions	2466
Table 1274. LINIBRR field descriptions	2467
Table 1275. LINCFR field descriptions	2468
Table 1276. LINC2R2 field descriptions	2469
Table 1277. BIDR field descriptions	2470
Table 1278. BDRL field descriptions	2471
Table 1279. BDRM field descriptions	2472
Table 1280. IFER field descriptions	2473
Table 1281. IFMI field descriptions	2474
Table 1282. IFMR field descriptions	2475
Table 1283. IFCR2n field descriptions	2476
Table 1284. IFCR2n+1 field descriptions	2477
Table 1285. GCR field descriptions	2478
Table 1286. UARTPTO field descriptions	2480
Table 1287. UARTCTO field descriptions	2481
Table 1288. DMATXE field descriptions	2482
Table 1289. DMARXE field descriptions	2483
Table 1290. LINFlexD register map	2484
Table 1291. Interrupts	2494
Table 1292. Interface signals	2496
Table 1293. MC_RGM register description	2503
Table 1294. MC_RGM memory map	2505
Table 1295. RGM_DES field descriptions	2515

Table 1296. RGM_DERD field descriptions	2517
Table 1297. RGM_DEAR field descriptions	2519
Table 1298. RGM_DBRE field descriptions	2520
Table 1299. RGM_FES field descriptions	2521
Table 1300. RGM_FERD field descriptions	2523
Table 1301. RGM_FEAR field descriptions	2525
Table 1302. RGM_FBRE field descriptions	2526
Table 1303. RGM_FESS field descriptions	2527
Table 1304. 'Functional' Reset Escalation Threshold Register (RGM_FRET) field descriptions	2528
Table 1305. RGM_DRET field descriptions	2529
Table 1306. RGM_EROC field descriptions	2530
Table 1307. Peripheral Reset Registers (RGM_PRSTn) field descriptions	2535
Table 1308. Peripheral Reset Registers (RGM_STATn) field descriptions	2542
Table 1309. MC_RGM reset implications	2542
Table 1310. MC_RGM alternate event selection	2547
Table 1311. BAF memory organization	2549
Table 1312. BAF Image Header	2549
Table 1313. BAF image version	2550
Table 1314. BAF Clock Jitter Activation Constant	2550
Table 1315. Remap configuration table	2553
Table 1316. BAF DCF client list	2555
Table 1317. SEC_SWT_CPU field description	2557
Table 1318. BAF DCF ADDR client description	2558
Table 1319. Serial boot callback function return value definition	2560
Table 1320. Locations of boot headers	2560
Table 1321. Boot header structure	2560
Table 1322. Boot header configuration field description	2561
Table 1323. Production Disable Callback routine entry point	2562
Table 1324. Production Disable protocol description	2562
Table 1325. MCAN_1 and LinFlexD_0 pin configuration	2569
Table 1326. LinFlex_0 and MCAN_1 baud rates	2570
Table 1327. UART serial boot mode download protocol	2570
Table 1328. Glossary	2572
Table 1329. SSCM memory map	2573
Table 1330. STATUS field descriptions	2574
Table 1331. MEMCONFIG field descriptions	2575
Table 1332. ERROR field descriptions	2576
Table 1333. UOPS field descriptions	2577
Table 1334. PSA field descriptions	2577
Table 1335. HSA field descriptions	2578
Table 1336. LCSTAT field descriptions	2579
Table 1337. DCF start record	2579
Table 1338. Format of a DCF data record	2580
Table 1339. DCF stop record	2580
Table 1340. Series of DCF records in UTEST	2580
Table 1341. Boot header structure	2582
Table 1342. Life Cycle Slots	2582
Table 1343. Life Cycle Slots in Memory	2583
Table 1344. Life Cycle Slots	2583
Table 1345. Power Management Controller Memory Map	2587
Table 1346. GR_S field descriptions	2589
Table 1347. GR_P field descriptions	2592

Table 1348. IE_P field descriptions	2595
Table 1349. EPR_VD3 field descriptions	2598
Table 1350. REE_VD3 field descriptions	2599
Table 1351. RES_VD3 field descriptions	2600
Table 1352. FEE_VD3 field descriptions	2602
Table 1353. EPR_VD4 field descriptions	2603
Table 1354. REE_VD4 field descriptions	2603
Table 1355. RES_VD4 field descriptions	2604
Table 1356. FEE_VD4 field descriptions	2605
Table 1357. EPR_VD7 field descriptions	2606
Table 1358. REE_VD7 field descriptions	2606
Table 1359. RES_VD7 field descriptions	2607
Table 1360. FEE_VD7 field descriptions	2608
Table 1361. EPR_VD8 field descriptions	2609
Table 1362. REE_VD8 field descriptions	2610
Table 1363. RES_VD8 field descriptions	2611
Table 1364. FEE_VD8 field descriptions	2612
Table 1365. EPR_VD9 field descriptions	2613
Table 1366. REE_VD9 field descriptions	2615
Table 1367. RES_VD9 field descriptions	2617
Table 1368. FEE_VD9 field descriptions	2617
Table 1369. EPR_VD10 field descriptions	2619
Table 1370. REE_VD10 field descriptions	2620
Table 1371. FEE_VD10 field descriptions	2621
Table 1372. EPR_VD12 field descriptions	2622
Table 1373. REE_VD12 field descriptions	2623
Table 1374. RES_VD12 field descriptions	2624
Table 1375. FEE_VD12 field descriptions	2625
Table 1376. EPR_VD13 field descriptions	2626
Table 1377. REE_VD13 field descriptions	2627
Table 1378. RES_VD13 field descriptions	2628
Table 1379. FEE_VD13 field descriptions	2629
Table 1380. EPR_VD14 field descriptions	2630
Table 1381. REE_VD14 field descriptions	2632
Table 1382. RES_VD14 field descriptions	2633
Table 1383. FEE_VD14 field descriptions	2634
Table 1384. EPR_VD15 field descriptions	2635
Table 1385. REE_VD15 field descriptions	2636
Table 1386. RES_VD15 field descriptions	2637
Table 1387. FEE_VD15 field descriptions	2638
Table 1388. VSIO field descriptions	2639
Table 1389. EPR_TD field descriptions	2640
Table 1390. REE_TD field descriptions	2641
Table 1391. RES_TD field descriptions	2642
Table 1392. CTL_TD field descriptions	2644
Table 1393. Temp Sensor FEE_TD field descriptions	2645
Table 1394. VD_UTST field descriptions	2646
Table 1395. ADC_CH Bit Descriptions	2647
Table 1396. VREG1P2_CTRL field descriptions	2647
Table 1397. MCR field descriptions	2648
Table 1398. MC_PCU registers	2654
Table 1399. Power Domain Status Register (PCU_PSTAT) Field Descriptions.	2655

Table 1400. MC_ME mode descriptions.	2658
Table 1401. MC_ME register description	2660
Table 1402. MC_ME memory map.	2664
Table 1403. Global Status Register (ME_GS) Field Descriptions	2673
Table 1404. Mode Control Register (ME_MCTL) Field Descriptions	2675
Table 1405. Mode Enable Register (ME_ME) Field Descriptions.	2676
Table 1406. Interrupt Status Register (ME_IS) Field Descriptions.	2677
Table 1407. Interrupt Mask Register (ME_IM) Field Descriptions	2678
Table 1408. Invalid Mode Transition Status Register (ME_IMTS) Field Descriptions	2680
Table 1409. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions	2681
Table 1410. Mode Configuration Registers (ME_<mode>_MC) Field Descriptions	2688
Table 1411. Peripheral Status Registers (ME_PSn) field descriptions.	2694
Table 1412. Run Peripheral Configuration Registers (ME_RUN_PC0...7) Field Descriptions.	2695
Table 1413. Low-Power Peripheral Configuration Registers (ME_LP_PC0...7) Field Descriptions.	2696
Table 1414. Peripheral Control Registers (ME_PCTLn) Field Descriptions	2697
Table 1415. Core Status Register (ME_CS) field descriptions.	2698
Table 1416. ME_CCTL0 field descriptions	2699
Table 1417. ME_CCTL1 field descriptions	2700
Table 1418. ME_CCTL2 field descriptions	2701
Table 1419. Core Control Register 3 (ME_CCTL3) Field Descriptions	2702
Table 1420. ME_CCTL4 field descriptions	2703
Table 1421. ME_CADDR0 field descriptions	2704
Table 1422. ME_CADDR1 field descriptions	2705
Table 1423. ME_CADDR3 field descriptions	2706
Table 1424. ME_CADDR4 field descriptions	2707
Table 1425. MC_ME resource control overview.	2713
Table 1426. MC_ME System Clock Selection Overview	2716
Table 1427. DAC events and resultant updates.	2731
Table 1428. DAC events and resultant updates, dual-issue case 1.	2733
Table 1429. DAC events and resultant updates, dual-issue case 2.	2735
Table 1430. DAC events and resultant updates, dual-issue case 3.	2736
Table 1431. DAC events and resultant updates, dual-issue case 4.	2737
Table 1432. DBCR0 field descriptions	2744
Table 1433. DBCR1 field descriptions	2746
Table 1434. DBCR2 field descriptions	2748
Table 1435. DBCR4 field descriptions	2752
Table 1436. DBCR5 field descriptions	2754
Table 1437. DBCR6 field descriptions	2756
Table 1438. DBCR7 field descriptions	2757
Table 1439. DBCR8 field descriptions	2759
Table 1440. DBSR field descriptions	2761
Table 1441. EDBRAC0 field descriptions	2764
Table 1442. EDBRAC0 resource control	2767
Table 1443. DEVENT field descriptions	2769
Table 1444. DDAM field descriptions	2770
Table 1445. EDBCR0 field descriptions	2774
Table 1446. EDBSR0 field descriptions	2775
Table 1447. EDBSRMSK0 field descriptions	2777
Table 1448. JTAG/OnCE primary interface signals	2781
Table 1449. OnCE Status register field descriptions	2785
Table 1450. OCMD field descriptions	2786
Table 1451. OnCE register addressing	2787

Table 1452. OCR field descriptions	2790
Table 1453. OnCE register access requirements.	2792
Table 1454. CTL field descriptions	2796
Table 1455. Watchpoint output signal assignments	2801
Table 1456. DAC events and resultant updates	2812
Table 1457. DAC events and resultant updates, dual-issue case 1	2814
Table 1458. DAC events and resultant updates, dual-issue case 2	2816
Table 1459. DAC events and resultant updates, dual-issue case 3	2817
Table 1460. DAC events and resultant updates, dual-issue case 4	2818
Table 1461. DBCR0 field descriptions	2825
Table 1462. DBCR1 field descriptions	2827
Table 1463. DBCR2 field descriptions	2829
Table 1464. DBCR4 field descriptions	2833
Table 1465. DBCR5 field descriptions	2835
Table 1466. DBCR6 field descriptions	2837
Table 1467. DBCR7 field descriptions	2838
Table 1468. DBCR8 field descriptions	2840
Table 1469. DBSR field descriptions	2842
Table 1470. EDBRAC0 field descriptions	2845
Table 1471. EDBRAC0 resource control	2848
Table 1472. DEVENT field descriptions	2851
Table 1473. DDAM field descriptions	2852
Table 1474. EDBCR0 field descriptions	2856
Table 1475. EDBSR0 field descriptions	2857
Table 1476. EDBSRMSK0 field descriptions	2859
Table 1477. JTAG/OnCE primary interface signals	2863
Table 1478. OnCE Status register field descriptions	2867
Table 1479. OCMD field descriptions	2868
Table 1480. OnCE register addressing	2869
Table 1481. OCR field descriptions	2872
Table 1482. OnCE register access requirements.	2874
Table 1483. CTL field descriptions	2878
Table 1484. Watchpoint output signal assignments	2883
Table 1485. DCI JTAG source selection	2890
Table 1486. DCI LFAST/JTAG port sharing	2891
Table 1487. DCI operating modes	2892
Table 1488. Nexus reset truth table	2898
Table 1489. DCI_CR field descriptions.	2903
Table 1490. MCU operation (JTAG to LFAST switching)	2905
Table 1491. Tool operation (JTAG to LFAST switching)	2905
Table 1492. MCU operation (LFAST to JTAG switching)	2906
Table 1493. Tool operation (LFAST to JTAG switching)	2906
Table 1494. MCU operation (LFAST to JTAGRST switching)	2906
Table 1495. Tool operation (LFAST to JTAGRST switching)	2907
Table 1496. JTAG signal properties	2910
Table 1497. Device identification register field descriptions	2912
Table 1498. JTAG_PASSWORD register field descriptions	2912
Table 1499. 6-bit JTAG instructions	2915
Table 1500. Abbreviations	2920
Table 1501. TAP.7 signal list	2923
Table 1502. Class/protocol relationship	2925
Table 1503. Reset state descriptions	2929

Table 1504. Reset effects	2930
Table 1505. Start-up Options	2930
Table 1506. Reset values for TAP.7 start-up options	2931
Table 1507. TAP.7 controller register list (managed with commands)	2937
Table 1508. Scan Format Register Values	2938
Table 1509. RDBACK register format	2942
Table 1510. Configuration Register 0 Format	2942
Table 1511. Reset values of TAP.7 controller registers managed with commands	2945
Table 1512. TAP.7 controller command list	2947
Table 1513. Store Miscellaneous Control	2949
Table 1514. Store Conditional 1 bit (STC1)	2949
Table 1515. Store Conditional 2 bit (STC2)	2950
Table 1516. Store Scan Format (STFMT)	2950
Table 1517. Make Conditional Group Member (MCM)	2950
Table 1518. Make Scan Group Candidate (MSC)	2951
Table 1519. Scan Bit (SCNB)	2951
Table 1520. Scan String (SCNS)	2952
Table 1521. ZBS use state characteristics with both TAP.7 and other ZBS use	2953
Table 1522. Factors determining the SP payload content	2965
Table 1523. MScan scan format use case summary	2966
Table 1524. MScan Scan Packet content	2967
Table 1525. OScan scan format use case summary	2969
Table 1526. OScan Scan Format "Payload"	2969
Table 1527. JDC memory map	2971
Table 1528. MCR register field descriptions	2971
Table 1529. MSR register field descriptions	2972
Table 1530. JOUT_IPS register field descriptions	2973
Table 1531. JIN_IPS register field descriptions	2973
Table 1532. JOUT JTAG register field descriptions	2974
Table 1533. JIN JTAG register field descriptions	2974
Table 1534. JTAG instructions	2975
Table 1535. SPU register summary	2983
Table 1536. ITLD register field descriptions	2990
Table 1537. C0PEVP register field descriptions	2991
Table 1538. C1PEVP register field descriptions	2992
Table 1539. C2PEVP register field descriptions	2992
Table 1540. C0PIS register field descriptions	2993
Table 1541. C1PIS register field descriptions	2994
Table 1542. C2PIS register field descriptions	2994
Table 1543. L2nSEL0 register field descriptions	2996
Table 1544. L2nSEL1 register field descriptions	2997
Table 1545. L2nSEL2 register field descriptions	2999
Table 1546. L2nSEL3 register field descriptions	3000
Table 1547. IOICn register field descriptions	3002
Table 1548. SCTRL register field descriptions	3003
Table 1549. SnGC register field descriptions	3006
Table 1550. SS register field descriptions	3007
Table 1551. SE register field descriptions	3009
Table 1552. SPU action decoder	3011
Table 1553. SnTA0 register field descriptions	3013
Table 1554. SnTA1 register field descriptions	3014
Table 1555. SnFA0 register field descriptions	3015

Table 1556. SnFA1 register field descriptions	3016
Table 1557. Trace group configuration register field descriptions	3017
Table 1558. INTS register field descriptions	3018
Table 1559. CCTRLn register field descriptions	3019
Table 1560. CCMPn register field descriptions	3020
Table 1561. CCOMS register field descriptions	3020
Table 1562. COS register field descriptions	3021
Table 1563. CCAPS register field descriptions	3022
Table 1564. Processor exception processing conditions	3024
Table 1565. Memory map	3038
Table 1566. JTAGM_MCR register field descriptions	3039
Table 1567. JTAGM_SR register field descriptions	3041
Table 1568. JTAGM_DOR0 register field descriptions	3043
Table 1569. JTAGM_DOR1 register field descriptions	3043
Table 1570. JTAGM_DOR2 register field descriptions	3044
Table 1571. JTAGM_DOR3 register field descriptions	3044
Table 1572. JTAGM_RxCRC register field descriptions	3045
Table 1573. JTAGM_DIR0 register field descriptions	3045
Table 1574. JTAGM_DIR1 register field descriptions	3046
Table 1575. Operating modes for LFAST interface	3048
Table 1576. Header payload sizes	3050
Table 1577. LFAST memory map	3051
Table 1578. Register conventions	3053
Table 1579. MCR field descriptions	3054
Table 1580. SCR field descriptions	3056
Table 1581. COCR field descriptions	3057
Table 1582. TMCr field descriptions	3058
Table 1583. ALCr field descriptions	3060
Table 1584. RCDcR field description	3061
Table 1585. SLcR field descriptions	3062
Table 1586. PICr field descriptions	3063
Table 1587. LcR field description	3064
Table 1588. UNSTcR field descriptions	3066
Table 1589. UNSTDRn field descriptions	3067
Table 1590. GSR field descriptions	3068
Table 1591. DFSR field descriptions	3069
Table 1592. TISR field descriptions	3070
Table 1593. RISR field descriptions	3071
Table 1594. RIISR field descriptions	3073
Table 1595. PLLLSR field descriptions	3074
Table 1596. UNSRSR field descriptions	3075
Table 1597. UNSRDRn field descriptions	3076
Table 1598. High speed 8 phase selection - sampling procedure	3081
Table 1599. High speed 4 phase selection - sampling procedure	3082
Table 1600. Low speed 4 phase selection - sampling procedure	3082
Table 1601. Line Receiver State	3086
Table 1602. Priority Levels for the Tranmsit Controller	3087
Table 1603. Line Driver States	3089
Table 1604. Frames supported by LFAST interfaces	3090
Table 1605. Loopback modes	3094
Table 1606. Frames Supported by LFAST interfaces	3100
Table 1607. Recommended receive exception handling mechanism	3101

Table 1608. Rx clocks summary	3106
Table 1609. Tx clocks summary	3106
Table 1610. DTS register access effects	3111
Table 1611. DTS memory map	3111
Table 1612. DTS_ENABLE register field descriptions	3112
Table 1613. DTS_STARTUP register field descriptions	3113
Table 1614. DTS_SEMAPHORE register field descriptions	3114
Table 1615. DTS_SEMAPHORE_B register field descriptions	3114
Table 1616. NAR registers	3119
Table 1617. NAR_CR field descriptions	3120
Table 1618. NAR_ST field descriptions	3122
Table 1619. NAR_SFR field descriptions	3123
Table 1620. NAR_TFR field descriptions	3125
Table 1621. NAR_TBAHI field descriptions	3126
Table 1622. NAR_TBALO field descriptions	3126
Table 1623. NAR_TCR field description	3127
Table 1624. NAR_STCR field descriptions	3129
Table 1625. NAR_CCSR field descriptions	3130
Table 1626. NAR_CDR field descriptions	3131
Table 1627. NAR_AHFPAR field description	3132
Table 1628. NAR operational modes truth table	3133
Table 1629. NAR transmit operations	3134
Table 1630. NAR message formats	3142
Table 1631. NAR error types	3143
Table 1632. Configuration error codes	3144
Table 1633. Watchpoint Hit values	3144
Table 1634. Implemented instructions	3148
Table 1635. Loading NEXUS-ENABLE instruction	3149
Table 1636. Writing a 32-bit value to a register	3150
Table 1637. Terms and Definitions	3154
Table 1638. Supported TCODEs	3159
Table 1639. Error Code (ECODE) Encoding (TCODE = 8)	3162
Table 1640. Error Type (ETYPE) Encoding (TCODE = 8)	3163
Table 1641. Resource Code (RCODE) values (TCODE = 27)	3163
Table 1642. Event Code (EVCODE) Encoding (TCODE = 33)	3163
Table 1643. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)	3163
Table 1644. Nexus 3 Register Map	3164
Table 1645. CSC field descriptions	3166
Table 1646. PCR field descriptions	3166
Table 1647. DC1 field descriptions	3167
Table 1648. DC2 field descriptions	3169
Table 1649. DC3 field descriptions	3171
Table 1650. DC4 field descriptions	3173
Table 1651. DS field descriptions	3174
Table 1652. WT field descriptions	3175
Table 1653. PTSTC field descriptions	3176
Table 1654. PTETC field descriptions	3177
Table 1655. DTSTC field descriptions	3178
Table 1656. DTETC field descriptions	3179
Table 1657. WMSK field descriptions	3181
Table 1658. DTC field descriptions	3183
Table 1659. Data trace — address range options	3186

Table 1660. RWCS field descriptions	3186
Table 1661. Read/Write Access Status Bit Encoding	3187
Table 1662. RWD data placement for transfers	3188
Table 1663. RWD byte lane mapping	3188
Table 1664. JTAG DR field descriptions	3190
Table 1665. Message Type Priority and Message Dropped Responses	3194
Table 1666. Indirect Branch Message sources	3198
Table 1667. Direct Branch Message sources	3198
Table 1668. Branch/Predicate history events	3201
Table 1669. RCODE encoding	3202
Table 1670. Program Trace exception summary	3204
Table 1671. Program Trace Exception Summary for w/Sync BTM messages	3206
Table 1672. Data Trace Exception summary	3210
Table 1673. Bus cycle special cases	3211
Table 1674. Watchpoint source encoding	3214
Table 1675. External Hardware Trigger Signals	3215
Table 1676. JTAG pins for Nexus 3	3221
Table 1677. Nexus 3 auxiliary pins	3221
Table 1678. Nexus port arbitration signals	3222
Table 1679. MSEO pin(s) protocol	3223
Table 1680. MDO request encodings	3226
Table 1681. Indirect branch message example (2 MDO / 1 MSEO)	3227
Table 1682. Indirect branch message example (8 MDO / 2 MSEO)	3227
Table 1683. Direct branch message example (2 MDO / 1 MSEO)	3228
Table 1684. Direct branch message example (8 MDO / 2 MSEO)	3228
Table 1685. Data write message example (8 MDO / 1 MSEO)	3228
Table 1686. Data write message example (8 MDO / 2 MSEO)	3229
Table 1687. Accessing internal Nexus 3 registers via JTAG/OnCE	3229
Table 1688. Accessing memory-mapped resources (reads)	3230
Table 1689. Accessing memory-mapped resources (writes)	3230
Table 1690. Terms and Definitions	3231
Table 1691. Supported TCODEs	3236
Table 1692. Error Code (ECODE) Encoding (TCODE = 8)	3239
Table 1693. Error Type (ETYPE) Encoding (TCODE = 8)	3240
Table 1694. Resource Code (RCODE) values (TCODE = 27)	3240
Table 1695. Event Code (EVCODE) Encoding (TCODE = 33)	3240
Table 1696. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)	3241
Table 1697. Nexus 3 Register Map	3241
Table 1698. CSC field descriptions	3243
Table 1699. PCR field descriptions	3243
Table 1700. DC1 field descriptions	3244
Table 1701. DC2 field descriptions	3246
Table 1702. DC3 field descriptions	3249
Table 1703. DC4 field descriptions	3251
Table 1704. DS field descriptions	3252
Table 1705. WT field descriptions	3253
Table 1706. PTSTC field descriptions	3254
Table 1707. PTETC field descriptions	3255
Table 1708. DTSTC field descriptions	3256
Table 1709. DTETC field descriptions	3257
Table 1710. WMSK field descriptions	3259
Table 1711. DTC field descriptions	3261

Table 1712. Data trace — address range options	3264
Table 1713. RWCS field descriptions	3264
Table 1714. Read/Write Access Status Bit Encoding	3265
Table 1715. RWD data placement for transfers	3266
Table 1716. RWD byte lane mapping	3266
Table 1717. JTAG DR field descriptions	3268
Table 1718. Message Type Priority and Message Dropped Responses	3272
Table 1719. Indirect Branch Message sources	3276
Table 1720. Direct Branch Message sources	3277
Table 1721. Branch/Predicate history events	3279
Table 1722. RCODE encoding	3280
Table 1723. Program Trace exception summary	3283
Table 1724. Program Trace Exception Summary for w/Sync BTM messages	3284
Table 1725. Data Trace Exception summary	3288
Table 1726. Bus cycle special cases	3289
Table 1727. Watchpoint source encoding	3293
Table 1728. External Hardware Trigger Signals	3294
Table 1729. JTAG pins for Nexus 3	3300
Table 1730. Nexus 3 auxiliary pins	3300
Table 1731. Nexus port arbitration signals	3301
Table 1732. MSEO pin(s) protocol	3302
Table 1733. MDO request encodings	3304
Table 1734. Indirect branch message example (2 MDO / 1 MSEO)	3305
Table 1735. Indirect branch message example (8 MDO / 2 MSEO)	3305
Table 1736. Direct branch message example (2 MDO / 1 MSEO)	3306
Table 1737. Direct branch message example (8 MDO / 2 MSEO)	3306
Table 1738. Data write message example (8 MDO / 1 MSEO)	3306
Table 1739. Data write message example (8 MDO / 2 MSEO)	3307
Table 1740. Accessing internal Nexus 3 registers via JTAG/OnCE	3307
Table 1741. Accessing memory-mapped resources (reads)	3308
Table 1742. Accessing memory-mapped resources (writes)	3308
Table 1743. GTM Sub-module instances	3311
Table 1744. GTMDI signal properties	3315
Table 1745. GTMDI registers	3316
Table 1746. DID field descriptions	3319
Table 1747. GTMDI_DC field descriptions	3320
Table 1748. GTMDI_DS field descriptions	3322
Table 1749. GTMDI_TIM_WPC1 field descriptions	3324
Table 1750. GTMDI_TIM_WPC2 field descriptions	3326
Table 1751. GTMDI_TOM_WPC1 field descriptions	3328
Table 1752. GTMDI_TOM_WPC2 field descriptions	3331
Table 1753. GTMDI_ATOM_WPC1 field descriptions	3333
Table 1754. GTMDI_ATOM_WPC2 field descriptions	3335
Table 1755. GTMDI_SPEA_WPC1 field descriptions	3337
Table 1756. GTMDI_SPEA_WPC2 field descriptions	3339
Table 1757. GTMDI_SPEB_WPC1 field descriptions	3341
Table 1758. GTMDI_SPEB_WPC2 field descriptions	3342
Table 1759. GTMDI_DPLL_WPC1 field descriptions	3344
Table 1760. GTMDI_DPLL_WPC2 field descriptions	3346
Table 1761. GTMDI_DPLL_WPC3 field descriptions	3347
Table 1762. GTMDI_DPLL_WPC4 field descriptions	3348
Table 1763. GTMDI_DPLL_WPC5 field descriptions	3349

Table 1764. GTMDI_DPLL_DTC field descriptions	3349
Table 1765. GTMDI_ARU_WPC1 field descriptions	3351
Table 1766. GTMDI_ARU_WPC2 field descriptions	3352
Table 1767. GTMDI_ARU_DATA0H field descriptions	3354
Table 1768. GTMDI_ARU_DATA0L field descriptions	3354
Table 1769. GTMDI_ARU_DATA1H field descriptions	3355
Table 1770. GTMDI_ARU_DATA1L field descriptions	3355
Table 1771. GTMDI_ARU_DTC field descriptions	3356
Table 1772. GTMDI_MCSA_DC and GTMDI_MCSB_DC field descriptions	3358
Table 1773. GTMDI_MCSA_WPC and GTMDI_MCSB_WPC field descriptions	3359
Table 1774. GTMDI_MCSA_PTC and GTMDI_MCSB_PTC field descriptions	3361
Table 1775. GTMDI_MCSA_DTC and GTMDI_MCSB_DTC field descriptions	3362
Table 1776. GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1 field descriptions	3363
Table 1777. GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 field descriptions	3364
Table 1778. GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1 field descriptions	3365
Table 1779. GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2 field descriptions	3365
Table 1780. GTMDI_MCSA_CE and GTMDI_MCSB_CE field descriptions	3366
Table 1781. GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 field descriptions	3368
Table 1782. GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2 field descriptions	3369
Table 1783. GTMDI_TBU0_WPC1 field descriptions	3370
Table 1784. GTMDI_TBU0_WPC2 field descriptions	3371
Table 1785. GTMDI_TBU0_DATA field descriptions	3373
Table 1786. GTMDI_TBU1_WPC1 field descriptions	3373
Table 1787. GTMDI_TBU1_WPC2 field descriptions	3375
Table 1788. GTMDI_TBU1_DATA field descriptions	3376
Table 1789. GTMDI_TBU2_WPC1 field descriptions	3377
Table 1790. GTMDI_TBU2_WPC2 field descriptions	3379
Table 1791. GTMDI_TBU2_DATA field descriptions	3380
Table 1792. GTMDI Nexus trace message formats	3383
Table 1793. Debug status field format	3385
Table 1794. Error field format	3385
Table 1795. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b000	3386
Table 1796. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b001	3386
Table 1797. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b010	3387
Table 1798. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b011	3387
Table 1799. Implemented instructions	3388
Table 1800. Loading NEXUS-ENABLE instruction	3390
Table 1801. Writing to a register	3391
Table 1802. JTAG pins for NXMC	3400
Table 1803. NXMC auxiliary pins	3400
Table 1804. Supported public TCODEs	3401
Table 1805. Data trace size (DSZ) encodings (TCODE = 5,6,13,14).	3402
Table 1806. Error Type (ETYPE) encodings (applicable to all error messages)	3402
Table 1807. Error Code (ECODE) encodings (applicable based on error source).	3403
Table 1808. NXMC registers	3404
Table 1809. DC1 register field description	3404
Table 1810. DC2 register field description	3406
Table 1811. WT register field description	3406
Table 1812. DTC field description	3407
Table 1813. Data trace address range options.	3408
Table 1814. BWC1 field description	3409
Table 1815. BWC2 field description	3410

Table 1816. JTAG DR field description for Nexus register access	3412
Table 1817. Data trace exception summary	3415
Table 1818. System Bus cycle special cases	3417
Table 1819. Watchpoint source description	3418
Table 1820. CRC unit memory map	3422
Table 1821. CRC_CFG field descriptions	3423
Table 1822. CRC_INP field descriptions	3424
Table 1823. CRC_CSTAT field descriptions	3425
Table 1824. CRC_OUTP field descriptions	3426
Table 1825. MEMU memory map	3437
Table 1826. CTRL field descriptions	3439
Table 1827. ERR_FLAG field descriptions	3440
Table 1828. DEBUG field descriptions	3443
Table 1829. SYS_RAM_CERR_STS _n field descriptions	3444
Table 1830. SYS_RAM_CERR_ADDR _n field descriptions	3445
Table 1831. SYS_RAM_UNCERR_STS field descriptions	3446
Table 1832. SYS_RAM_UNCERR_ADDR field descriptions	3446
Table 1833. SYS_RAM_OFLW _n field descriptions	3447
Table 1834. PERIPH_RAM_CERR_STS _n field descriptions	3448
Table 1835. PERIPH_RAM_CERR_ADDR _n field descriptions	3448
Table 1836. PERIPH_RAM_UNCERR_STS field descriptions	3449
Table 1837. PERIPH_RAM_UNCERR_ADDR field descriptions	3450
Table 1838. PERIPH_RAM_OFLW _n field descriptions	3451
Table 1839. FLASH_CERR_STS _n register field descriptions	3451
Table 1840. FLASH_CERR_ADDR _n register field descriptions	3452
Table 1841. UNCERR_STS field descriptions	3453
Table 1842. FLASH_UNCERR_ADDR field descriptions	3453
Table 1843. FLASH_OFLW _n field descriptions	3454
Table 1844. CDACNTL field descriptions	3459
Table 1845. CDADATA field descriptions	3460
Table 1846. CDACNTL field descriptions	3461
Table 1847. IMA Memory Map	3464
Table 1848. IMA_CTRL field descriptions	3465
Table 1849. IMA_ENABLE field descriptions	3466
Table 1850. IMA_STATUS field descriptions	3467
Table 1851. IMA_SLCT field descriptions	3468
Table 1852. IMA_WRITE_UNLOCK field descriptions	3469
Table 1853. IMA_READ_UNLOCK field descriptions	3470
Table 1854. IMA_WRITE_DATA _n field descriptions	3470
Table 1855. IMA_READ_DATA _n field descriptions	3471
Table 1856. Acronyms and abbreviations	3476
Table 1857. FCCU submodules	3477
Table 1858. Input reset	3481
Table 1859. Reset reactions	3481
Table 1860. Reset sources	3481
Table 1861. MC actions	3481
Table 1862. FCCU error pin behavior on STATE transitions	3488
Table 1863. Dual-rail encoding	3488
Table 1864. Time-switching encoding	3489
Table 1865. Bi-stable encoding	3490
Table 1866. FCCU register set	3493
Table 1867. FCCU_CTRL field descriptions	3496

Table 1868. FCCU_CTRLK field descriptions	3498
Table 1869. FCCU_CFG field descriptions	3499
Table 1870. FCCU_RF_CFG[0:3] field descriptions	3501
Table 1871. FCCU RF configuration register channels	3501
Table 1872. FCCU_RFS_CFG[0:7] field descriptions	3502
Table 1873. FCCU RFS configuration register channels	3502
Table 1874. FCCU_RF_S[0:3] field descriptions	3504
Table 1875. FCCU RF status register channels	3504
Table 1876. FCCU_RFK field descriptions	3505
Table 1877. FCCU_RF_E[0:3] field descriptions	3506
Table 1878. FCCU RFE enable register channels	3506
Table 1879. FCCU_RF_TOE[0:3] field descriptions	3507
Table 1880. FCCU RF timeout enable register channels	3507
Table 1881. FCCU_RF_TO field descriptions	3508
Table 1882. FCCU_CFG_TO field descriptions	3509
Table 1883. Bi-stable encoding	3509
Table 1884. FCCU_EINOUT field descriptions	3510
Table 1885. FCCU_STAT field descriptions	3511
Table 1886. N2AF_STATUS field descriptions	3512
Table 1887. A2FF_STATUS field descriptions	3514
Table 1888. N2FF_STATUS field descriptions	3515
Table 1889. F2AF_STATUS field descriptions	3516
Table 1890. FCCU_RFF field descriptions	3517
Table 1891. FCCU_IRQ_STAT field descriptions	3518
Table 1892. FCCU_IRQ_EN field descriptions	3519
Table 1893. Timer state/value	3519
Table 1894. FCCU_XTMR field descriptions	3520
Table 1895. FCCU_MCS field descriptions	3521
Table 1896. FCCU_TRANS_LOCK field descriptions	3521
Table 1897. FCCU_PERMNT_LOCK field descriptions	3522
Table 1898. FCCU_DELTA_T field descriptions	3523
Table 1899. FCCU_IRQ_ALARM_EN[0:3] field descriptions	3523
Table 1900. FCCU IRQ alarm enable register channels	3523
Table 1901. FCCU_NMI_EN[0:3] field descriptions	3524
Table 1902. FCCU NMI enable register channels	3524
Table 1903. FCCU_EOUT_SIG_EN[0:3] field descriptions	3525
Table 1904. FCCU EOUT signaling enable register channels	3525
Table 1905. FCCU register map	3526
Table 1906. STCU2 register set	3540
Table 1907. STCU_RUN field descriptions	3542
Table 1908. STCU_RUNSW field descriptions	3544
Table 1909. STCU_SKC field descriptions	3545
Table 1910. STCU_CFG field descriptions	3547
Table 1911. STCU_PLL_CFG field descriptions	3549
Table 1912. STCU_WDG field descriptions	3550
Table 1913. STCU_INT_FLG field descriptions	3551
Table 1914. STCU_CRCE field descriptions	3552
Table 1915. STCU_CRCR field descriptions	3552
Table 1916. STCU_ERR_STAT field descriptions	3553
Table 1917. STCU_ERR_FM field descriptions	3556
Table 1918. STCU_LBS field descriptions	3557
Table 1919. STCU_LBE field descriptions	3558

Table 1920. STCU_LBSSW field descriptions	3559
Table 1921. STCU_LBESW field descriptions	3559
Table 1922. STCU_LBRMSW field descriptions	3560
Table 1923. STCU_LBUFM field descriptions	3561
Table 1924. STCU_MBSL field descriptions	3562
Table 1925. STCU_MBSM field descriptions	3562
Table 1926. STCU_MBSH field descriptions	3563
Table 1927. STCU_MBEL field descriptions	3564
Table 1928. STCU_MBEM field descriptions	3565
Table 1929. STCU_MBEH field descriptions	3565
Table 1930. STCU_MBSLSW field descriptions	3566
Table 1931. STCU_MBSMSW field descriptions	3567
Table 1932. STCU_MBSHSW field descriptions	3568
Table 1933. STCU_MBELSW field descriptions	3569
Table 1934. STCU_MBEMSW field descriptions	3570
Table 1935. STCU_MBEHSW field descriptions	3571
Table 1936. STCU_MBUFML field descriptions	3572
Table 1937. STCU_MBUFMM field descriptions	3573
Table 1938. STCU_MBUFMMH field descriptions	3574
Table 1939. STCU_LB_CTRL field descriptions	3575
Table 1940. STCU_LB_PCS field descriptions	3577
Table 1941. STCU_LB_PRPGL field descriptions	3578
Table 1942. STCU_LB_PRPGH field descriptions	3579
Table 1943. STCU_LB_MISREL field descriptions	3580
Table 1944. STCU_LB_MISREH field descriptions	3581
Table 1945. STCU_LB_MISRRL field descriptions	3581
Table 1946. STCU_LB_MISRRLH field descriptions	3582
Table 1947. STCU_LB_MISRELSW field descriptions	3583
Table 1948. STCU_LB_MISREHSW field descriptions	3584
Table 1949. STCU_LB_MISRRLSW field descriptions	3585
Table 1950. STCU_LB_MISRRLHSW field descriptions	3586
Table 1951. STCU_MB_CTRL field descriptions	3587
Table 1952. STCU2 register map	3588
Table 1953. REG_PROT Memory Map	3600
Table 1954. SLBR _n field descriptions	3601
Table 1955. Soft Lock Bits vs. Protected Address	3602
Table 1956. GCR field descriptions	3603
Table 1957. Module memory map	3610
Table 1958. LCSTAT register field descriptions	3612
Table 1959. CHSEL register field descriptions	3613
Table 1960. CHSEL register field descriptions	3614
Table 1961. CIN register field descriptions	3615
Table 1962. Production Disable DCF Client bit definition	3617
Table 1963. Production Disable DCF Client bit definition	3618
Table 1964. LOCK0_PG _n register field descriptions	3619
Table 1965. LOCK1_PG _n register field descriptions	3621
Table 1966. LOCK2_PG _n register field descriptions	3622
Table 1967. LOCK3_PG _n register field descriptions	3624
Table 1968. Security features mapped to life cycle	3625
Table 1969. Possible life cycle records	3626
Table 1970. Moving life cycle to “OEM Production”	3627
Table 1971. Moving life cycle to <i>In the Field</i>	3627

Table 1972. CS/ADDR Values for Lock Registers	3627
Table 1973. Setting some bits in the MIDLOCK field	3628
Table 1974. Truth Table of Production Disable Feature	3630
Table 1975. DCF client address map	3632
Table 1976. TDM IPS registers address map	3633
Table 1977. DBA field descriptions	3634
Table 1978. TO field descriptions	3635
Table 1979. OTPEN0 field descriptions	3636
Table 1980. OTPEN1 field descriptions	3637
Table 1981. OTPEN2 field descriptions	3637
Table 1982. OTPEN3 field descriptions	3638
Table 1983. TDRn_LOCK0 field descriptions	3639
Table 1984. TDRn_LOCK1 field descriptions	3640
Table 1985. TDRn_LOCK2 field descriptions	3641
Table 1986. TDRn_LOCK3 field descriptions	3642
Table 1987. TDRSR field descriptions	3643
Table 1988. LFPAR field descriptions	3643
Table 1989. AMU memory map	3649
Table 1990. VCHn field description	3652
Table 1991. UCHn field description	3652
Table 1992. P1CHn field description	3653
Table 1993. P2CHn field description	3654
Table 1994. P3CHn field description	3654
Table 1995. P8CHn field description	3655
Table 1996. P6CHn field description	3656
Table 1997. NCHn field description	3656
Table 1998. P7CHn field description	3657
Table 1999. P4CHn, P5CHn field description	3658
Table 2000. RESCHn field description	3658
Table 2001. V32_n field description	3659
Table 2002. V64Mn field description	3660
Table 2003. DSACHn field description	3661
Table 2004. DDACHn field description	3662
Table 2005. DCNTCHn field description	3662
Table 2006. DCTLCHn field description	3663
Table 2007. STCFGn field description	3665
Table 2008. ESR field description	3668
Table 2009. DVHCHn field description	3669
Table 2010. Data types in registers	3671
Table 2011. WKPU memory map	3677
Table 2012. NSR field descriptions	3679
Table 2013. NCR field descriptions	3680
Table 2014. High level memory map	3688
Table 2015. GTMINT Module Memory Map	3690
Table 2016. GTMMCR field descriptions	3710
Table 2017. GTMINTCLR field descriptions	3711
Table 2018. Correspondence between the INTCLR_PTR and the interrupts outputs	3711
Table 2019. GTMMAEICR field descriptions	3713
Table 2020. Submodules of GTM-IP	3715
Table 2021. ECC Characteristics for GTM Memories	3716
Table 2022. 42 bits Word Based Syndrome Definition	3717
Table 2023. JTAG operating frequencies	3727

Table 2024. Reset sources and functional blocks	3727
Table 2025. BD memory map	3728
Table 2026. e2eECC Basic H-Matrix Definition	3731
Table 2027. BD_SIUL2 register summary	3739
Table 2028. BD_SIUL2_DIDR1 Bit Field Descriptions	3740
Table 2029. BD_SIUL2_DIDR2 Bit Field Descriptions	3741
Table 2030. BD_SIUL2_ISR Bit Field Descriptions	3742
Table 2031. BD_SIUL2_RSR Bit Field Descriptions	3743
Table 2032. DCI JTAG source selection	3747
Table 2033. DCI LFAST/JTAG port sharing	3748
Table 2034. DCI operating modes	3749
Table 2035. Nexus reset truth table	3756
Table 2036. DCI_CR field descriptions	3761
Table 2037. MCU operation (JTAG to LFAST switching)	3763
Table 2038. Tool operation (JTAG to LFAST switching)	3763
Table 2039. MCU operation (LFAST to JTAG switching)	3763
Table 2040. Tool operation (LFAST to JTAG switching)	3764
Table 2041. MCU operation (LFAST to JTAGRST switching)	3764
Table 2042. Tool operation (LFAST to JTAGRST switching)	3765
Table 2043. JTAG signal properties	3768
Table 2044. Device identification register field descriptions	3770
Table 2045. JTAG_PASSWORD register field descriptions	3770
Table 2046. 6-bit JTAG instructions	3773
Table 2047. Memory map	3785
Table 2048. JTAGM_MCR register field descriptions	3786
Table 2049. JTAGM_SR register field descriptions	3788
Table 2050. JTAGM_DOR0 register field descriptions	3790
Table 2051. JTAGM_DOR1 register field descriptions	3790
Table 2052. JTAGM_DOR2 register field descriptions	3791
Table 2053. JTAGM_DOR3 register field descriptions	3791
Table 2054. JTAGM_RxCRC register field descriptions	3792
Table 2055. JTAGM_DIR0 register field descriptions	3792
Table 2056. JTAGM_DIR1 register field descriptions	3793
Table 2057. NAR registers	3796
Table 2058. NAR_CR field descriptions	3797
Table 2059. NAR_ST field descriptions	3799
Table 2060. NAR_SFR field descriptions	3801
Table 2061. NAR_TFR field descriptions	3803
Table 2062. NAR_TBAHI field descriptions	3804
Table 2063. NAR_TBALO field descriptions	3804
Table 2064. NAR_TCR field description	3805
Table 2065. NAR_STCR field descriptions	3807
Table 2066. NAR_CCSR field descriptions	3808
Table 2067. NAR_CDR field descriptions	3809
Table 2068. NAR_NAPCR field descriptions	3810
Table 2069. NAR_AHFPAR field description	3811
Table 2070. NAR operational modes truth table	3813
Table 2071. NAR transmit operations	3814
Table 2072. NAR message formats	3822
Table 2073. NAR error types	3823
Table 2074. Configuration error codes	3824
Table 2075. Watchpoint Hit values	3825

Table 2076. Implemented instructions	3828
Table 2077. Loading NEXUS-ENABLE instruction	3829
Table 2078. Writing a 32-bit value to a register	3830
Table 2079. NAL and NAP feature summary	3835
Table 2080. NAL memory map	3836
Table 2081. NAL_GSR field descriptions	3837
Table 2082. NAL_GCR field descriptions	3838
Table 2083. NAL_TCR field descriptions	3840
Table 2084. NAP signals	3844
Table 2085. CSPU encoding	3849
Table 2086. Standard DSZ encoding	3850
Table 2087. NXMC DSZ encoding	3851
Table 2088. Address translation example	3852
Table 2089. Message field description	3856
Table 2090. JTAG DR field description for DWPU register access	3857
Table 2091. Tag RAM configuration	3858
Table 2092. System address to Tag address translation	3859
Table 2093. JTAG DWPU registers	3859
Table 2094. Memory mapped DWPU registers	3861
Table 2095. GLB_CTRL field descriptions	3861
Table 2096. SRC_MAP_CTLRn field descriptions	3862
Table 2097. FAR_SA_CFGn field descriptions	3863
Table 2098. FAR_RS_CFGn field descriptions	3864
Table 2099. TRC_CTRL field descriptions	3864
Table 2100. TRC_ADR field descriptions	3865
Table 2101. TRC_DATA field descriptions	3866
Table 2102. ORC_CTRL1 field descriptions	3866
Table 2103. ORC_CTRL2 field descriptions	3868
Table 2104. GLB_STAT field descriptions	3869
Table 2105. ORC_CFGn_ADR field descriptions	3870
Table 2106. ORC_CFGn_DH field descriptions	3870
Table 2107. ORC_CFGn_DL field descriptions	3871
Table 2108. ORC_CFGn_DM field descriptions	3871
Table 2109. RWA register map	3872
Table 2110. RWCS fields	3873
Table 2111. Read/Write access status bit encoding	3874
Table 2112. Protection encoding	3874
Table 2113. RWD data placement for transfers	3875
Table 2114. RWD byte lane mapping	3875
Table 2115. JTAG DR field description for Nexus register access	3876
Table 2116. Operating modes for LFAST interface	3880
Table 2117. Header payload sizes	3882
Table 2118. LFAST memory map	3883
Table 2119. Register conventions	3885
Table 2120. MCR field descriptions	3886
Table 2121. SCR field descriptions	3888
Table 2122. COCR field descriptions	3889
Table 2123. TMCR field descriptions	3890
Table 2124. ALCR field descriptions	3892
Table 2125. RCDCR field description	3893
Table 2126. SLCR field descriptions	3894
Table 2127. PICR field descriptions	3895

Table 2128. PLLCR field descriptions	3896
Table 2129. LCR field description	3898
Table 2130. UNSTCR field descriptions	3900
Table 2131. UNSTDR _n field descriptions	3901
Table 2132. GSR field descriptions	3902
Table 2133. DFSSR field descriptions	3903
Table 2134. TISR field descriptions	3904
Table 2135. RISR field descriptions	3905
Table 2136. RIISR field descriptions	3907
Table 2137. PLLLSR field descriptions	3908
Table 2138. UNSRSR field descriptions	3909
Table 2139. UNSRDR _n field descriptions	3910
Table 2140. High speed 8 phase selection - sampling procedure	3916
Table 2141. High speed 4 phase selection - sampling procedure	3917
Table 2142. Low speed 4 phase selection - sampling procedure	3917
Table 2143. Line Receiver State	3921
Table 2144. Priority Levels for the Transmitter Controller	3922
Table 2145. Line Driver States	3924
Table 2146. Frames supported by LFAST interfaces	3925
Table 2147. Loopback modes	3929
Table 2148. Frames Supported by LFAST interfaces	3936
Table 2149. Recommended receive exception handling mechanism	3936
Table 2150. Rx clocks summary	3942
Table 2151. Tx clocks summary	3942
Table 2152. Document revision history	3944

List of figures

Figure 1.	Key to Register Fields.	154
Figure 2.	Block diagram.	164
Figure 3.	Periphery allocation	165
Figure 4.	Standby RAM Regulator Block Diagram.	168
Figure 5.	SPC57EM80xx flash memory block diagram	170
Figure 6.	Flash memory segmentation and read-while-write partitioning.	172
Figure 7.	General view of E2E ECC	206
Figure 8.	Dual core concept.	228
Figure 9.	Crossbar switch diagram	239
Figure 10.	Computational shell's crossbar switch integration	240
Figure 11.	Crossbar switch integration	242
Figure 12.	SMPU0 configuration	244
Figure 13.	SMPU1 configuration	246
Figure 14.	Peripheral bridge configuration.	248
Figure 15.	FEC Burst Optimization Master Control Register (FBOMCR)	249
Figure 16.	FBOMCR field description	249
Figure 17.	IAHB Burst Enable 1 (IAHB_BE1) Register	250
Figure 18.	IAHB Burst Enable 2 (IAHB_BE2) Register	252
Figure 19.	Interrupt timing	294
Figure 20.	DMA controller configuration	295
Figure 21.	DMA channel multiplexing	297
Figure 22.	PIT configuration.	308
Figure 23.	LFAST and SIPI block diagram	312
Figure 24.	DSPI configuration	313
Figure 25.	DSPI Module Configuration Register (DSPIx_MCR)	316
Figure 26.	Stopwatch timer source muxing	317
Figure 27.	MEMU system connections	323
Figure 28.	PASS_LOCK0_PGn Register.	345
Figure 29.	PASS_LOCK1_PGn Register.	346
Figure 30.	PASS_LOCK2_PGn Register.	347
Figure 31.	PASS_LOCK3_PGn Register.	348
Figure 32.	Tamper Region Override (TO) DCF client	351
Figure 33.	TDRx_LOCK0/OTPEN0 DCF client	352
Figure 34.	TDRx_LOCK1/OTPEN1 DCF client	353
Figure 35.	TDRx_LOCK2/OTPEN2 DCF client	354
Figure 36.	TDRx_LOCK3/OTPEN3 DCF client	354
Figure 37.	Debug and calibration architecture for SPC57EM80xx ED.	358
Figure 38.	Reset Generation Module Reset Sequence	363
Figure 39.	Boot-up sequence part A	387
Figure 40.	Boot-up sequence part B	388
Figure 41.	Boot-up sequence, part C.	389
Figure 42.	Boot-up sequence, part D.	390
Figure 43.	Boot-up sequence, part E.	391
Figure 44.	DCF record structure	393
Figure 45.	DCF start record	394
Figure 46.	DCF stop record	395
Figure 47.	Appending DCF records	396
Figure 48.	UTEST Miscellaneous DCF Client	415

Figure 49.	PMC_REE bits DCF Client	416
Figure 50.	Mode relationships	421
Figure 51.	Device power management framework	422
Figure 52.	SPC57EM80xx PMU overview	423
Figure 53.	Internal LVD and HVD in configuration example during power-up	427
Figure 54.	VDD_LV monitored voltages during power-up	436
Figure 55.	Example of VDD_LV power-up sequencing	438
Figure 56.	Threshold variation during power-up sequence: hot point LVD enabled	438
Figure 57.	Threshold variation during power-up sequence: hot point LVD disabled	439
Figure 58.	VDD_HV monitored voltages during power-up.	440
Figure 59.	Threshold variation during power-up sequence — 2.7V–5.5V hot point LVD disabled, 4.5V–5.5V range	441
Figure 60.	DCI EVTx pin multiplexing control register (DCI_PINCR).	448
Figure 61.	JTAG hierarchical view	465
Figure 62.	Aurora interface block diagram	467
Figure 63.	NAR block diagram	468
Figure 64.	NAP block diagram	471
Figure 65.	NXMC integration block diagram	473
Figure 66.	e200z block diagram	478
Figure 67.	e200z420n3 Supervisor Mode Programmer's Model SPRs	485
Figure 68.	e200z420n3 Supervisor Mode Programmer's Model DCRs and PMRs	486
Figure 69.	e200z420n3 User Mode Programmer's Model SPRs	487
Figure 70.	e200z420n3 User Mode Programmer's Model PMRs	487
Figure 71.	EFPU Status and Control Register (SPEFSCR).	488
Figure 72.	L1 Cache Control and Status Register 0 (L1CSR0)	491
Figure 73.	L1 Cache Control and Status Register 1 (L1CSR1)	494
Figure 74.	L1 Cache Configuration Register 0 (L1CFG0)	496
Figure 75.	L1 Cache Configuration Register 1 (L1CFG1)	497
Figure 76.	L1 Flush/Invalidate Register 0 (L1FINV0).	499
Figure 77.	L1 Flush/Invalidate Register 1 (L1FINV1).	499
Figure 78.	Exception Syndrome Register (ESR)	503
Figure 79.	Machine State Register (MSR)	504
Figure 80.	Machine Check Syndrome Register (MCSR)	506
Figure 81.	Zen Interrupt Vector Prefix Register (IVPR)	508
Figure 82.	MMU/MPU Configuration Register (MMUCFG)	520
Figure 83.	MPU0 Configuration Register (MPU0CFG)	520
Figure 84.	MPU0 Control and Status Register 0 (MPU0CSR0).	521
Figure 85.	MPU Assist Register 0 (MAS0).	524
Figure 86.	MPU Assist Register 1 (MAS1).	527
Figure 87.	MPU Assist Register 2 (MAS2).	528
Figure 88.	MPU Assist Register 3 (MAS3).	528
Figure 89.	MPU Assist Registers Summary	529
Figure 90.	DMEM Configuration Register 0 (DMEMCFG0).	530
Figure 91.	DMEM Control Register 0 (DMEMCTL0).	531
Figure 92.	DMEM Control Register 1 (DMEMCTL1).	534
Figure 93.	IMEM Configuration Register 0 (IMEMCFG0)	538
Figure 94.	IMEM Control Register 0 (IMEMCTL0)	539
Figure 95.	IMEM Control Register 1 (IMEMCTL1)	541
Figure 96.	e2eECC Control Register 0 (E2ECTL0)	544
Figure 97.	e2eECC Error Control/Status Register 0 (E2EECSR0)	545
Figure 98.	e200z425n3 Supervisor Mode Programmer's Model SPRs	548
Figure 99.	e200z425n3 Supervisor Mode Programmer's Model DCRs and PMRs	549

Figure 100.	e200z425n3 User Mode Programmer's Model SPRs	550
Figure 101.	e200z425n3 User Mode Programmer's Model PMRs	550
Figure 102.	LSP/EFPU Status and Control Register (SPEFSCR)	552
Figure 103.	L1 Cache Control and Status Register 0 (L1CSR0)	555
Figure 104.	L1 Cache Control and Status Register 1 (L1CSR1)	555
Figure 105.	L1 Cache Configuration Register 0 (L1CFG0)	557
Figure 106.	L1 Cache Configuration Register 1 (L1CFG1)	558
Figure 107.	L1 Flush/Invalidate Register 1 (L1FINV1)	559
Figure 108.	Exception Syndrome Register (ESR)	562
Figure 109.	Machine State Register (MSR)	563
Figure 110.	Machine Check Syndrome Register (MCSR)	565
Figure 111.	Zen Interrupt Vector Prefix Register (IVPR)	567
Figure 112.	MMU/MPU Configuration Register (MMUCFG)	579
Figure 113.	MPU0 Configuration Register (MPU0CFG)	579
Figure 114.	MPU0 Control and Status Register 0 (MPU0CSR0)	580
Figure 115.	MPU Assist Register 0 (MAS0)	583
Figure 116.	MPU Assist Register 1 (MAS1)	586
Figure 117.	MPU Assist Register 2 (MAS2)	587
Figure 118.	MPU Assist Register 3 (MAS3)	587
Figure 119.	MPU Assist Registers Summary	588
Figure 120.	DMEM Configuration Register 0 (DMEMCFG0)	589
Figure 121.	DMEM Control Register 0 (DMEMCTL0)	590
Figure 122.	DMEM Control Register 1 (DMEMCTL1)	593
Figure 123.	IMEM Configuration Register 0 (IMEMCFG0)	597
Figure 124.	IMEM Control Register 0 (IMEMCTL0)	598
Figure 125.	IMEM Control Register 1 (IMEMCTL1)	600
Figure 126.	e2eECC Control Register 0 (E2ECTL0)	603
Figure 127.	e2eECC Error Control/Status Register 0 (E2EECSR0)	604
Figure 128.	SIUL2 block diagram	607
Figure 129.	SIUL2 MCU ID Register #1 (SIUL2_MIDR1)	611
Figure 130.	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	612
Figure 131.	SIUL2 DMA/Interrupt Status Flag Register (SIUL2_DISR0)	613
Figure 132.	SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)	614
Figure 133.	SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)	615
Figure 134.	SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREEER0)	616
Figure 135.	SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	617
Figure 136.	SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)	618
Figure 137.	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0)	619
Figure 138.	SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)	620
Figure 139.	SIUL2 SoC Configuration Register0 (SIUL2_SCR0)	620
Figure 140.	MSCR I/O Pin and IP Block Port Connectivity	621
Figure 141.	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0 - SIUL2_MSCR511)	622
Figure 142.	Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512 - SIUL2_MSCR1023)	625
Figure 143.	SIUL2 GPIO Pad Data Output register 0_3 (SIUL2_GPDO0_3)	626
Figure 144.	SIUL2 GPIO Pad Data Input register 0_3 (SIUL2_GPDI0_3)	627
Figure 145.	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPD00)	627
Figure 146.	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDIO)	628
Figure 147.	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPD00)	629
Figure 148.	Data Port example arrangement showing configuration for different port width accesses	631
Figure 149.	External interrupt pad diagram	632

Figure 150.	Interrupt to vector mapping at MCU level	634
Figure 151.	Priority Registers Slave (XBARx_PRSn)	637
Figure 152.	XBAR Control Register (XBARx_CRSn)	640
Figure 153.	XBIC system block diagram	645
Figure 154.	XBIC block diagram	646
Figure 155.	XBIC Module Control Register (XBIC_MCR)	647
Figure 156.	XBIC Error Injection Register (XBIC_EIR)	648
Figure 157.	XBIC Error Status Register (XBIC_ESR)	649
Figure 158.	Error Address Register (XBIC_EAR)	650
Figure 159.	CLC block diagram	653
Figure 160.	Crossbar Lock Control Global Enable (CLC_GEN) register	654
Figure 161.	Crossbar Lock Control Configuration Register (CLC_MEN)	655
Figure 162.	Crossbar Lock Control Slave Target (CLC_STGT) register	657
Figure 163.	Master Privilege Register A (MPRA)	664
Figure 164.	Master Privilege Register B (MPRB)	664
Figure 165.	MPROT n fields	664
Figure 166.	Peripheral Access Control Registers (AIPS_PACRA–H)	666
Figure 167.	PACR n fields	666
Figure 168.	Off-Platform Peripheral Access Control Registers (AIPS_OPACRA–AF)	668
Figure 169.	OPACR n fields	669
Figure 170.	SMPU block diagram	671
Figure 171.	Control/Error Status Register 0 (SMPUx_CESR0)	677
Figure 172.	Control/Error Status Register 1 (SMPUx_CESR1)	678
Figure 173.	Error Address Register, Bus Master n (SMPUx_EAR n)	679
Figure 174.	Error Detail Register, Bus Master n (SMPUx_EDR n)	680
Figure 175.	Region Descriptor n , Word 0 (SMPUx_RGD n _WORD0)	681
Figure 176.	Region Descriptor n , Word 1 (SMPUx_RGD n _WORD1)	681
Figure 177.	Region Descriptor n , Word 2 Format 0 (SMPUx_RGD n _WORD2_FMT0)	682
Figure 178.	Region Descriptor n , Word 3 (SMPUx_RGD n _WORD3)	684
Figure 179.	SMPU access evaluation macro	685
Figure 180.	Read, pending read (1:1 timing mode)	690
Figure 181.	Write, pending read (1:1 timing mode)	690
Figure 182.	Burst4 read (1:1 timing mode)	691
Figure 183.	Burst4 write (1:1 timing mode)	691
Figure 184.	Read, pending read (2:1 timing mode)	692
Figure 185.	Write, pending read (2:1 timing mode)	693
Figure 186.	Burst4 read (2:1 timing mode)	694
Figure 187.	Burst4 write (2:1 timing mode)	695
Figure 188.	Read, pending read (1:2 timing mode)	696
Figure 189.	Write, pending read (1:2 timing mode)	697
Figure 190.	Burst4 read (1:2 timing mode)	698
Figure 191.	Burst4 write (1:2 timing mode)	699
Figure 192.	Semaphores2 block diagram	703
Figure 193.	Semaphore2 Gate Register (GATE)	704
Figure 194.	Semaphore2 Reset Gate Register (RSTGT)	706
Figure 195.	SEMA4_GATE n state machine	707
Figure 196.	INTC block diagram	710
Figure 197.	INTC Block Configuration Register (INTC_BCR)	714
Figure 198.	INTC Master Protection Register (INTC_MPROT)	714
Figure 199.	INTC Current Priority Register for Processor n (INTC_CPR n)	716
Figure 200.	INTC Interrupt Acknowledge Register for Processor n (INTC_IACKR n)	717
Figure 201.	INTC End of Interrupt Register for Processor n (INTC_EOIR n)	718

Figure 202.	INTC Software Set/Clear Interrupt Register 0–3 (INTC_SS CIR0_3)	719
Figure 203.	INTC Priority Select Register 0 (INTC_PSR0)	720
Figure 204.	INTC Priority Select Register 1023 (INTC_PSR1023)	720
Figure 205.	Timing diagram of software vector mode handshaking	725
Figure 206.	Timing diagram for hardware vector mode handshaking	726
Figure 207.	OSEK Code Sequence	732
Figure 208.	Interrupt request block diagram	732
Figure 209.	Timing diagram of raised priority preserved	733
Figure 210.	eDMA block diagram	737
Figure 211.	Control Register (DMA_CR)	784
Figure 212.	Error Status Register (DMA_ES)	786
Figure 213.	Enable Request Register High (DMA_ERQH)	788
Figure 214.	Enable Request Register Low (DMA_ERQL)	791
Figure 215.	Enable Error Interrupt Register High (DMA_EEIH)	794
Figure 216.	Enable Error Interrupt Register Low (DMA_EEIL)	797
Figure 217.	Set Enable Request Register (DMA_SERQ)	800
Figure 218.	Clear Enable Request Register (DMA_CERQ)	800
Figure 219.	Set Enable Error Interrupt Register (DMA_SEEI)	801
Figure 220.	Clear Enable Error Interrupt Register (DMA_CEEI)	802
Figure 221.	Clear Interrupt Request Register (DMA_CINT)	802
Figure 222.	Clear Error Register (DMA_CERR)	803
Figure 223.	Set START Bit Register (DMA_SSRT)	804
Figure 224.	Clear DONE Status Bit Register (DMA_CDNE)	804
Figure 225.	Interrupt Request Register High (DMA_INTH)	805
Figure 226.	Interrupt Request Register Low (DMA_INTL)	809
Figure 227.	Error Register High (DMA_ERRH)	812
Figure 228.	Error Register Low (DMA_ERRL)	815
Figure 229.	Hardware Request Status Register High (DMA_HRSH)	818
Figure 230.	Hardware Request Status Register Low (DMA_HRSL)	821
Figure 231.	Channel <i>n</i> Priority Register (DMA_DCHPRIn)	824
Figure 232.	Channel <i>n</i> Master ID Register (DMA_DCHMIDn)	825
Figure 233.	TCD Source Address (DMA_TCDn_SADDR)	825
Figure 234.	TCD Transfer Attributes (DMA_TCDn_ATTR)	826
Figure 235.	TCD Signed Source Address Offset (DMA_TCDn_SOFF)	827
Figure 236.	TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCDn_NBYTES_MLNO)	828
Figure 237.	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCDn_NBYTES_MLOFFNO)	829
Figure 238.	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCDn_NBYTES_MLOFFYES)	829
Figure 239.	TCD Last Source Address Adjustment (DMA_TCDn_SLAST)	830
Figure 240.	TCD Destination Address (DMA_TCDn_DADDR)	831
Figure 241.	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_CITER_ELINKYES)	832
Figure 242.	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_CITER_ELINKNO)	833
Figure 243.	TCD Signed Destination Address Offset (DMA_TCDn_DOFF)	833
Figure 244.	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCDn_DLASTSGA)	834
Figure 245.	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_BITER_ELINKYES)	835
Figure 246.	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_BITER_ELINKNO)	836

Figure 247. TCD Control and Status (DMA_TCDn_CSR)	837
Figure 248. DMA operation, part 1.	841
Figure 249. DMA operation, part 2.	842
Figure 250. DMA operation, part 3.	843
Figure 251. Example of multiple loop iterations.	850
Figure 252. Memory array terms	851
Figure 253. DMA Mux Block Diagram	862
Figure 254. Channel Configuration Registers (CHCONFIG#n)	864
Figure 255. DMA Mux triggered channels	866
Figure 256. DMA Mux Channel Triggering: Normal Operation	866
Figure 257. DMA Mux Channel Triggering: Ignored Trigger	867
Figure 258. Clock generation (part 1 of 2)	875
Figure 259. Clock generation (part 2 of 2)	876
Figure 260. SPC57EM80xx Dual PLL digital interface block diagram	882
Figure 261. Clock distribution	887
Figure 262. Device LFAST clocking.	889
Figure 263. Device Ethernet clocking	890
Figure 264. Sigma-Delta ADC clocking	891
Figure 265. SYSCLK/EXTCLK/REF_CLK pin muxing.	892
Figure 266. CMU0 block diagram	893
Figure 267. CMU[1:10] block diagram	893
Figure 268. Loss of clock logic.	896
Figure 269. PLL0 Control Register (PLL0CR)	899
Figure 270. PLL0 Status Register (PLL0SR)	900
Figure 271. PLL0 Divider Register (PLL0DV)	901
Figure 272. PLL1 Control Register (PLL1CR)	903
Figure 273. PLL1 Status Register (PLL1SR)	904
Figure 274. PLL1 Divider Register (PLL1DV)	905
Figure 275. PLL1 Frequency Modulation Register (PLL1FM)	907
Figure 276. PLL1 Fractional Divider Register (PLL1FD)	908
Figure 277. Triangular frequency modulation	911
Figure 278. Clock Monitor Unit diagram	915
Figure 279. CMU Control Status Register (CMU_CSR)	917
Figure 280. CMU Frequency Display Register (CMU_FDR)	918
Figure 281. CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	919
Figure 282. CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	919
Figure 283. CMU Interrupt Status Register (CMU_ISR)	920
Figure 284. CMU Measurement Duration Register (CMU_MDR)	921
Figure 285. MC_CGM block diagram	924
Figure 286. Auxiliary Clock 5 Cascaded Divider 0 Configuration Register (CGM_AC5_CDC0)	941
Figure 287. Auxiliary Clock 5 Cascaded Divider 1 Configuration Register (CGM_AC5_CDC1)	941
Figure 288. Auxiliary Clock 5 Cascaded Divider 2 Configuration Register (CGM_AC5_CDC2)	942
Figure 289. Auxiliary Clock 5 Cascaded Divider 10 Configuration Register (CGM_AC5_CDC10)	943
Figure 290. Auxiliary Clock 5 Cascaded Divider 11 Configuration Register (CGM_AC5_CDC11)	943
Figure 291. Auxiliary Clock 5 Cascaded Divider 12 Configuration Register (CGM_AC5_CDC12)	944
Figure 292. Auxiliary Clock 5 Cascaded Divider 13 Configuration Register (CGM_AC5_CDC13)	945
Figure 293. Auxiliary Clock 5 Cascaded Divider 20 Configuration Register (CGM_AC5_CDC20)	946
Figure 294. Auxiliary Clock 5 Cascaded Divider 21 Configuration Register (CGM_AC5_CDC21)	946
Figure 295. Auxiliary Clock 5 Cascaded Divider 22 Configuration Register (CGM_AC5_CDC22)	947
Figure 296. Auxiliary Clock 5 Cascaded Divider 23 Configuration Register (CGM_AC5_CDC23)	948
Figure 297. PCS Switch Duration Register (CGM_PCS_SDUR)	948
Figure 298. PCS Divider Change Register 1 (CGM_PCS_DIVC1)	949

Figure 299. PCS Divider Start Register 1 (CGM_PCS_DIVS1)	950
Figure 300. PCS Divider End Register 1 (CGM_PCS_DIVE1)	950
Figure 301. PCS Divider Change Register 2 (CGM_PCS_DIVC2)	951
Figure 302. PCS Divider Start Register 2 (CGM_PCS_DIVS2)	952
Figure 303. PCS Divider End Register 2 (CGM_PCS_DIVE2)	952
Figure 304. PCS Divider Change Register 4 (CGM_PCS_DIVC4)	953
Figure 305. PCS Divider Start Register 4 (CGM_PCS_DIVS4)	954
Figure 306. PCS Divider End Register 4 (CGM_PCS_DIVE4)	954
Figure 307. System Clock Select Status Register (CGM_SC_SS)	955
Figure 308. System Clock Divider 0 Configuration Register (CGM_SC_DC0)	957
Figure 309. System Clock Divider 1 Configuration Register (CGM_SC_DC1)	957
Figure 310. System Clock Divider 2 Configuration Register (CGM_SC_DC2)	958
Figure 311. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)	959
Figure 312. Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)	960
Figure 313. Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)	961
Figure 314. Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)	961
Figure 315. Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)	962
Figure 316. Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)	963
Figure 317. Auxiliary Clock 0 Divider 4 Configuration Register (CGM_AC0_DC4)	964
Figure 318. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)	964
Figure 319. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)	965
Figure 320. Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)	966
Figure 321. Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)	967
Figure 322. Auxiliary Clock 2 Divider 1 Configuration Register (CGM_AC2_DC1)	968
Figure 323. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)	968
Figure 324. Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)	969
Figure 325. Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)	970
Figure 326. Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)	971
Figure 327. Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)	972
Figure 328. Auxiliary Clock 5 Divider 1 Configuration Register (CGM_AC5_DC1)	973
Figure 329. Auxiliary Clock 5 Divider 2 Configuration Register (CGM_AC5_DC2)	974
Figure 330. Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)	975
Figure 331. Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)	976
Figure 332. Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)	977
Figure 333. Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)	977
Figure 334. Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)	978
Figure 335. Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)	979
Figure 336. Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)	980
Figure 337. Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)	981
Figure 338. Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)	982
Figure 339. Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)	983
Figure 340. Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)	984
Figure 341. Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)	985
Figure 342. Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)	986
Figure 343. Auxiliary Clock 10 Select Status Register (CGM_AC10_SS)	987
Figure 344. Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)	988
Figure 345. MC_CGM system clock generation overview	990
Figure 346. MC_CGM System Clock Ramp-Down Timing (k = 6 example)	992
Figure 347. MC_CGM System Clock Ramp-Up Timing (k = 6 example)	993
Figure 348. MC_CGM auxiliary clock 0 generation overview	996
Figure 349. MC_CGM auxiliary clock 1 generation overview	997
Figure 350. MC_CGM auxiliary clock 2 generation overview	997

Figure 351. MC_CGM auxiliary clock 3 generation overview	998
Figure 352. MC_CGM auxiliary clock 4 generation overview	998
Figure 353. MC_CGM auxiliary clock 5 generation overview	999
Figure 354. MC_CGM auxiliary clock 6 generation overview	1000
Figure 355. MC_CGM auxiliary clock 7 generation overview	1000
Figure 356. MC_CGM auxiliary clock 8 generation overview	1001
Figure 357. MC_CGM auxiliary clock 9 generation overview	1001
Figure 358. MC_CGM auxiliary clock 10 generation overview	1002
Figure 359. MC_CGM auxiliary clock 11 generation overview	1002
Figure 360. MC_CGM fractional division example waveform (divide by 2.3).	1006
Figure 361. XOSC_CTL register	1010
Figure 362. Control register (CTL)	1013
Figure 363. Native Trimming Register (NT)	1014
Figure 364. Temperature Trimming Register (TT)	1015
Figure 365. Simplified platform block diagram	1016
Figure 366. Platform RAM Configuration Register 1 (PRCR1)	1017
Figure 367. Read-modify-write datapath	1020
Figure 368. RAM controller transaction monitor block diagram	1025
Figure 369. Platform-centric simple block diagram with flash memory controller	1027
Figure 370. PFCR1 register	1030
Figure 371. PFCR2 register	1034
Figure 372. PFCR3 register	1037
Figure 373. PFAPR register	1040
Figure 374. PFCRCR register	1043
Figure 375. PFCRDE register	1045
Figure 376. PFlash Calibration Region Descriptor <i>n</i> , Word0 (PFCRD <i>n</i> .Word0)	1052
Figure 377. PFlash Calibration Region Descriptor <i>n</i> , Word1 (PFCRD <i>n</i> .Word1)	1053
Figure 378. PFlash Calibration Region Descriptor <i>n</i> , Word2 (PFCRD <i>n</i> .Word2)	1053
Figure 379. Interlock Write Alternate Programming Interface Routing Flowchart	1058
Figure 380. Port0 Access Followed By Port1 Access(RWSC=5, APC=0)	1059
Figure 381. Port0 Access Followed By Port1 Access(RWSC=5, APC=1)	1059
Figure 382. Port0 Access Followed By Port1 Access(RWSC=5, APC=2)	1060
Figure 383. Port0 Access Followed By Port1 Access(RWSC=5, APC=3)	1060
Figure 384. Port0 Access Followed By Port1 Access(RWSC=5, APC=3'b1xx)	1061
Figure 385. Flash memory controller 4-entry, 2-way mini-cache organization.	1061
Figure 386. PFlash calibration remap connection matrix	1064
Figure 387. PFlash calibration remap logic detail	1066
Figure 388. Flash memory controller and associated calibration remap targets	1068
Figure 389. Safety-critical calibration remap datapath with redundancy	1072
Figure 390. Block diagram of c55fmc	1074
Figure 391. MCR Register	1084
Figure 392. MCRa Register	1089
Figure 393. MCRE Register	1090
Figure 394. LOCK0 register	1093
Figure 395. LOCK1 Register	1097
Figure 396. LOCK2 Register	1099
Figure 397. LOCK3 Register	1100
Figure 398. LOCK0a Register	1101
Figure 399. LOCK1a Register	1104
Figure 400. SEL0 Register	1106
Figure 401. SEL1 Register	1107
Figure 402. SEL2 Register	1108

Figure 403. SEL3 Register	1109
Figure 404. ADR Register	1110
Figure 405. UT0 Register	1112
Figure 406. UM0 Register	1115
Figure 407. UM1 Register	1116
Figure 408. UM2 Register	1117
Figure 409. UM3 Register	1118
Figure 410. UM4 Register	1119
Figure 411. UM5 Register	1120
Figure 412. UM6 Register	1121
Figure 413. UM7 Register	1122
Figure 414. UM8 Register	1123
Figure 415. UM9 Register	1124
Figure 416. OPP0 Register	1126
Figure 417. OPP1 Register	1127
Figure 418. OPP2 Register	1128
Figure 419. OPP3 Register	1129
Figure 420. TMD Register	1130
Figure 421. Power Architecture Decorated Load and Store Instruction Formats	1146
Figure 422. DSMC Block Diagram	1148
Figure 423. Decoration Format: BFINS	1149
Figure 424. Decoration Format: CAST	1151
Figure 425. Decoration Format: AND	1151
Figure 426. Decoration Format: OR	1152
Figure 427. Decoration Format: XOR	1152
Figure 428. Decoration Format: SLD	1153
Figure 429. Decoration Format: SWAP	1153
Figure 430. Decoration Format: LAS1	1154
Figure 431. DSMC Timing Diagram	1155
Figure 432. Flash memory protection layers	1156
Figure 433. SEL0 flash control register	1158
Figure 434. LOCK0 flash control register	1161
Figure 435. Secure write protection	1163
Figure 436. DCF record structure	1164
Figure 437. Sample DCF record for secure write protection	1166
Figure 438. LOCK0_PGn register	1167
Figure 439. PASS_LOCK3_PGn Register	1170
Figure 440. Data portion of Tamper Region Override (TO) DCF record	1176
Figure 441. SPC57EM80xx ADC subsystem	1186
Figure 442. Sigma-Delta ADC integration diagram	1189
Figure 443. Sigma-Delta ADC clock diagram	1190
Figure 444. Sigma-Delta ADC simultaneous start connectivity	1191
Figure 445. Sigma-Delta ADC external modulator interface	1192
Figure 446. SAR ADC integration diagram	1202
Figure 447. SAR ADC clock diagram	1203
Figure 448. SARADC input trigger diagram	1204
Figure 449. Self Test implementation	1206
Figure 450. Analog input pin pullup/pulldown pad cell	1207
Figure 451. Analog input AN35 — discharge switch to AVSS	1207
Figure 452. SARADC_B ADC read of internal PMC voltages	1210
Figure 453. SDADC block diagram	1249
Figure 454. Module Configuration Register (MCR)	1253

Figure 455. Channel Selection Register (CSR)	1256
Figure 456. Reset Key Register (RKR)	1258
Figure 457. Status Flag Register (SFR).	1259
Figure 458. Request Select and Enable Register (RSER)	1261
Figure 459. Output Settling Delay Register (OSDR)	1262
Figure 460. FIFO Control Register (FCR)	1263
Figure 461. Software Trigger Key Register (STKR).	1264
Figure 462. Converted Data Register (CDR)	1265
Figure 463. WDG Threshold Register (WTHHLR).	1266
Figure 464. SARADC block diagram	1274
Figure 465. Normal conversion flow	1276
Figure 466. Injected sample/conversion sequence	1277
Figure 467. Analog conversion sequence	1279
Figure 468. Analog watchdog guarded area	1281
Figure 469. Main Configuration Register (MCR)	1287
Figure 470. Main Status Register (MSR)	1290
Figure 471. Interrupt Status Register (ISR)	1291
Figure 472. Internal Channel Interrupt Pending Registers (ICIPR0–2)	1292
Figure 473. Interrupt Mask Register (IMR)	1294
Figure 474. Internal Channel Interrupt Mask Registers (ICIMR0–2)	1295
Figure 475. Watchdog Threshold Interrupt Status Register (WTISR)	1296
Figure 476. Watchdog Threshold Interrupt Mask Register (WTIMR).	1297
Figure 477. DMA Enable Register (DMAE)	1298
Figure 478. Internal Channel DMA Select Registers (ICDSR0–2).	1299
Figure 479. Watchdog Threshold Registers 0–3 (WTHRHLR0–3)	1300
Figure 480. Conversion Timing Registers 0–3 (CTR0–3)	1301
Figure 481. Internal Channel Normal Conversion Mask Registers (ICNCMR0–2)	1302
Figure 482. Internal Channel Injected Conversion Mask Registers (ICJCMR0–2)	1303
Figure 483. Power Down Exit Delay Register (PDEDL)	1304
Figure 484. Internal Channel Data Registers (ICDR0–95)	1305
Figure 485. Watchdog Threshold Registers 4–15 (WTHRHLR4–15)	1306
Figure 486. Internal Channel Watchdog Select Registers (ICWSELR0–11)	1307
Figure 487. Internal Channel Watchdog Enable Registers (ICWENR0–2)	1308
Figure 488. Internal Channel Analog Watchdog Out of Range Registers (ICAWORR0–2).	1309
Figure 489. Test Channel Interrupt Pending Register (TCIPR)	1311
Figure 490. Test Channel Interrupt Mask Registers (TCIMR)	1312
Figure 491. Test Channel DMA Select Register (TCDSR)	1313
Figure 492. Test Channel Normal Conversion Mask Registers (TCNCMR)	1314
Figure 493. Test Channel Injected Conversion Mask Register (TCJCMR)	1315
Figure 494. Test Channels Watchdog Select Registers (TCWSELR)	1316
Figure 495. Test Channel Watchdog Enable Register (TCWENR)	1317
Figure 496. Test Channel Analog Watchdog Out of Range Register (TCAWORR)	1318
Figure 497. Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–7)	1319
Figure 498. Test Channel Data Registers (TCDR96–127)	1320
Figure 499. External Channel Decode Signals Delay Register (ECDSDR).	1322
Figure 500. External Channel Interrupt Pending Register (ECIPR0–3).	1323
Figure 501. External Channel Interrupt Mask Registers (ECIMR0–3)	1324
Figure 502. External Channel DMA Select Registers (ECDSR0–3)	1325
Figure 503. External Channel Normal Conversion Mask Registers (IECNCMR0–3).	1326
Figure 504. External Channel Injected Conversion Mask Registers (ECJCMR0–3)	1327
Figure 505. External Channel Watchdog Select Registers (ECWSELR0–15).	1328
Figure 506. External Channel Watchdog Enable Registers (ECWENR0–3)	1329

Figure 507. External Channel Analog Watchdog Out of Range Registers (ECAWORR0–3)	1330
Figure 508. External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–3)	1331
Figure 509. External Channel Data Registers (ECDR128–255)	1332
Figure 510. Digital output behavior with temperature	1337
Figure 511. Temperature formula	1338
Figure 512. Calibration Points	1339
Figure 513. STM Control Register (STM_CR)	1343
Figure 514. STM Count Register (STM_CNT)	1344
Figure 515. STM Channel Control Register (STM_CCRn)	1345
Figure 516. STM Channel Interrupt Register (STM_CIRn)	1345
Figure 517. STM Channel Compare Register (STM_CMPn)	1346
Figure 518. SWT Control Register (SWT_CR)	1348
Figure 519. SWT Interrupt Register (SWT_IR)	1350
Figure 520. SWT Time-Out Register (SWT_TO)	1350
Figure 521. SWT Window Register (SWT_WN)	1351
Figure 522. SWT Service Register (SWT_SR)	1351
Figure 523. SWT Counter Output Register (SWT_CO)	1352
Figure 524. SWT Service Key Register (SWT_SK)	1352
Figure 525. Pseudorandom Key Generator	1354
Figure 526. Block diagram of the PIT	1356
Figure 527. PIT Module Control Register (PITx_MCR)	1359
Figure 528. PIT Upper Lifetime Timer Register (PITx_LTMR64H)	1360
Figure 529. PIT Lower Lifetime Timer Register (PITx_LTMR64L)	1361
Figure 530. PIT Module x Timer Load Value Register n (PITx_LDVALn)	1361
Figure 531. PIT Module x Current Timer Value Register n (PITx_CVALn)	1362
Figure 532. Timer Control Register (PITx_TCTRLn)	1363
Figure 533. Timer Flag Register n (PITx_TFLGn)	1364
Figure 534. PIT Module x Timer Load Value Register n (PITx_LDVALn)	1364
Figure 535. PIT Module x Current Timer Value Register n (PITx_CVALn)	1365
Figure 536. Timer Control Register (PITx_TCTRLn)	1366
Figure 537. Timer Flag Register n (PITx_TFLGn)	1367
Figure 538. Stopping and starting a timer	1368
Figure 539. Modifying running timer period	1368
Figure 540. Dynamically setting a new load value	1368
Figure 541. CAN subsystem generic block diagram	1373
Figure 542. M_CAN core block diagram	1376
Figure 543. Core Release Register	1380
Figure 544. Endian Register	1381
Figure 545. Fast Bit Timing and Prescaler Register	1382
Figure 546. Fast Bit Timing and Prescaler Register	1382
Figure 547. Test Register	1383
Figure 548. RAM watchdog register	1384
Figure 549. CC Control register	1385
Figure 550. Bit Timing and Prescaler Register	1387
Figure 551. Timestamp Counter Configuration Register	1389
Figure 552. Timestamp counter value register	1390
Figure 553. Timeout counter configuration register	1391
Figure 554. Timeout counter value register	1392
Figure 555. Error counter register	1392
Figure 556. Protocol status register (PSR)	1393
Figure 557. Interrupt register	1396
Figure 558. Interrupt enable register	1399

Figure 559. Interrupt line select register	1402
Figure 560. Interrupt line enable register	1405
Figure 561. Global filter configuration register	1406
Figure 562. Standard ID Filter Configuration register	1407
Figure 563. Extended ID Filter Configuration register	1408
Figure 564. Extended ID and Mask register	1409
Figure 565. High Priority Message Status register	1410
Figure 566. NDAT1 register	1411
Figure 567. NDAT1 register field descriptions	1411
Figure 568. NDAT2 register	1411
Figure 569. NDAT2 register field descriptions	1412
Figure 570. Rx FIFO 0 Configuration register	1412
Figure 571. Rx FIFO 0 Acknowledge register	1414
Figure 572. Rx FIFO 0 Configuration register	1415
Figure 573. Rx FIFO 0 Configuration register	1415
Figure 574. Rx FIFO 1 Status register	1416
Figure 575. Rx FIFO 1 Acknowledge register	1417
Figure 576. Tx Buffer Configuration register	1418
Figure 577. Tx FIFO/Queue Status register	1419
Figure 578. Tx Buffer Transmission Occurred register	1422
Figure 579. Tx Buffer Cancellation Finished register	1423
Figure 580. Tx Buffer Transmission Interrupt Enable register	1423
Figure 581. Tx Buffer Cancellation Finished Interrupt Enable register	1424
Figure 582. Tx Event FIFO Configuration register	1425
Figure 583. Tx Event FIFO Status register	1426
Figure 584. Tx Event FIFO Acknowledge register	1427
Figure 585. Message RAM Configuration	1428
Figure 586. Rx FIFO Element	1428
Figure 587. Tx Buffer Element	1430
Figure 588. Tx Event FIFO Element	1431
Figure 589. Standard Message ID Filter Element	1432
Figure 590. Extended Message ID Filter Element	1434
Figure 591. Transceiver delay measurement	1440
Figure 592. Pin control in Bus Monitoring mode	1441
Figure 593. External Loop Back mode	1443
Figure 594. Internal loop back mode	1444
Figure 595. Standard Message ID Filter Path	1447
Figure 596. Extended Message ID Filter Path	1448
Figure 597. Debug Message Handling State Machine	1451
Figure 598. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO	1453
Figure 599. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue	1453
Figure 600. Timing of DMA interface signals	1455
Figure 601. M_TTCAN core block diagram	1457
Figure 602. Core Release register	1462
Figure 603. Endian Register	1463
Figure 604. Fast Bit Timing and Prescaler Register	1464
Figure 605. Test Register	1465
Figure 606. RAM watchdog register	1466
Figure 607. CC Control register	1467
Figure 608. Bit Timing and Prescaler Register	1470
Figure 609. Timestamp Counter Configuration Register	1471
Figure 610. Timestamp counter value register	1472

Figure 611. Timeout counter configuration register	1473
Figure 612. Timeout counter value register	1474
Figure 613. Error counter register	1474
Figure 614. Protocol status register (PSR)	1475
Figure 615. Interrupt register	1478
Figure 616. Interrupt enable register	1481
Figure 617. Interrupt line select register	1484
Figure 618. Interrupt line enable register	1487
Figure 619. Global filter configuration register	1488
Figure 620. Standard ID Filter Configuration register	1489
Figure 621. Extended ID Filter Configuration register	1490
Figure 622. Extended ID and Mask register	1491
Figure 623. High Priority Message Status register	1492
Figure 624. NDAT1 register	1493
Figure 625. NDAT1 register field descriptions	1493
Figure 626. NDAT2 register	1493
Figure 627. NDAT2 register field descriptions	1494
Figure 628. Rx FIFO 0 Configuration register	1494
Figure 629. Rx FIFO 0 Status register	1495
Figure 630. Rx FIFO 0 Acknowledge register	1496
Figure 631. Rx FIFO 0 Configuration register	1496
Figure 632. Rx FIFO 1 Configuration register	1497
Figure 633. Rx FIFO 1 Status register	1498
Figure 634. Rx FIFO 1 Acknowledge register	1499
Figure 635. Tx Buffer Configuration register	1499
Figure 636. Tx FIFO/Queue Status register	1500
Figure 637. Tx Buffer Request Pending register	1501
Figure 638. Tx Buffer Add Request register	1502
Figure 639. Tx Buffer Cancellation Request register	1503
Figure 640. Tx Buffer Transmission Occurred register	1504
Figure 641. Tx Buffer Cancellation Finished register	1504
Figure 642. Tx Buffer Transmission Interrupt Enable register	1505
Figure 643. Tx Buffer Cancellation Finished Interrupt Enable register	1506
Figure 644. Tx Event FIFO Configuration register	1506
Figure 645. Tx Event FIFO Status register	1507
Figure 646. Tx Event FIFO Acknowledge register	1508
Figure 647. TT Trigger Memory Configuration register	1509
Figure 648. TT Reference Message Configuration register	1510
Figure 649. TT Operation Configuration register	1511
Figure 650. TT Matrix Limits register	1512
Figure 651. TUR Configuration register	1514
Figure 652. TT Operation Control register	1515
Figure 653. TT Global Time Preset register	1517
Figure 654. TT Time Mark register	1518
Figure 655. TT Interrupt Register	1519
Figure 656. TT Interrupt Enable register	1521
Figure 657. TT Interrupt Line Select register	1523
Figure 658. TT Operation Status register	1525
Figure 659. TUR Numerator Actual register	1527
Figure 660. TT Local and Global Time register	1528
Figure 661. TT Cycle Time and Count register	1528
Figure 662. TT Capture Time register	1529

Figure 663. TT Cycle Sync Mark register	1530
Figure 664. Message RAM Configuration	1531
Figure 665. Rx Buffer and FIFO Element	1532
Figure 666. Tx Buffer Element	1533
Figure 667. Tx Event FIFO Element	1534
Figure 668. Standard Message ID Filter Element	1536
Figure 669. Extended Message ID Filter Element	1537
Figure 670. Trigger Memory Element	1538
Figure 671. Transceiver delay measurement	1542
Figure 672. Pin Control in Bus Monitoring Mode	1543
Figure 673. Pin Control in Loop Back modes	1546
Figure 674. Standard Message ID Filter Path	1549
Figure 675. Extended Message ID Filter Path	1550
Figure 676. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO	1553
Figure 677. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue	1554
Figure 678. First byte of Level 1 reference message	1556
Figure 679. First four bytes of Level 2 reference message	1556
Figure 680. First four bytes of Level 0 reference message	1557
Figure 681. TUR configuration examples	1558
Figure 682. Cycle Time and Global Time Synchronization	1567
Figure 683. M_TTCAN Level 0 and Level 2 Drift Compensation	1568
Figure 684. Level 0 schedule synchronization state machine	1575
Figure 685. Level 0 master to slave relation	1576
Figure 686. Hardware control for M_TTCAN cycles management	1581
Figure 687. M_CAN_1 and M_CAN_2 connections to GP I/Os	1582
Figure 688. Interprocessor communication diagram	1586
Figure 689. SIPI module	1586
Figure 690. SIPI block diagram	1587
Figure 691. SIPI register read request	1589
Figure 692. SIPI read response	1589
Figure 693. SIPI register write request	1589
Figure 694. SIPI write acknowledge	1589
Figure 695. Streaming write request format ^{(1),(2)}	1590
Figure 696. SIPI header coding	1590
Figure 697. SIPI header channel field	1591
Figure 698. Read request transfer	1594
Figure 699. Read answer transfer	1594
Figure 700. Data allocation	1595
Figure 701. Register write transfer	1596
Figure 702. Write transfer	1596
Figure 703. Streaming transfer with data	1597
Figure 704. Write Acknowledge transfer	1598
Figure 705. ID request transfer	1598
Figure 706. ID request response transfer	1599
Figure 707. SIPI single register write API	1599
Figure 708. SIPI Single Register Write API – Flow Chart	1600
Figure 709. SIPI Multiple Register Write API	1601
Figure 710. SIPI Single Register Read API	1602
Figure 711. SIPI multiple register read API	1603
Figure 712. SIPI data stream model	1604
Figure 713. SIPI data stream model - Initiator	1604
Figure 714. SIPI data stream model - Target	1605

Figure 715. SIPI data stream with acknowledge trigger	1605
Figure 716. Interrupt description	1609
Figure 717. SIPI control & status overview – Register transfer	1610
Figure 718. Channel Control Register (CCRn)	1612
Figure 719. Channel Status Register (CSRn)	1614
Figure 720. Channel Interrupt Register (CIRn)	1615
Figure 721. Channel Timeout Register (CTORn)	1616
Figure 722. Channel CRC Register (CCRCn)	1617
Figure 723. Channel address Register (CARn)	1618
Figure 724. Channel Data Register (CDRn)	1618
Figure 725. SIPI Module Configuration Register (SIPI_MCR)	1619
Figure 726. SIPI Status Register (SIPI_SR)	1621
Figure 727. Max Count Register (MAXCR)	1622
Figure 728. SIPI Address Reload Register (SIPIARR)	1623
Figure 729. SIPI Address Count Register (SIPIACR)	1624
Figure 730. SIPI Channel Error Register (SIPIERR)	1625
Figure 731. DSPI block diagram	1627
Figure 732. DSPI with queues and DMA	1630
Figure 733. DSPI Module Configuration Register (DSPIx_MCR)	1637
Figure 734. DSPI Transfer Count Register (DSPIx_TCR)	1640
Figure 735. DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPIx_CTARn)	1641
Figure 736. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPIx_CTARn_SLAVE)	1646
Figure 737. DSPI Status Register (DSPIx_SR)	1648
Figure 738. DSPI DMA/Interrupt Request Select and Enable Register (DSPIx_RSER)	1651
Figure 739. DSPI PUSH FIFO Register In Master Mode (DSPIx_PUSHR)	1654
Figure 740. DSPI PUSH FIFO Register In Slave Mode (DSPIx_PUSHR_SLAVE)	1657
Figure 741. DSPI POP FIFO Register (DSPIx_POPR)	1658
Figure 742. DSPI Transmit FIFO Registers (DSPIx_TXFRn)	1659
Figure 743. DSPI Receive FIFO Registers (DSPIx_RXFRn)	1660
Figure 744. DSPI DSI Configuration Register 0 (DSPIx_DSICR0)	1661
Figure 745. DSPI DSI Serialization Data Register 0 (DSPIx_SDR0)	1664
Figure 746. DSPI DSI Alternate Serialization Data Register 0 (DSPIx_ASDR0)	1665
Figure 747. DSPI DSI Transmit Comparison Register 0 (DSPIx_COMPR0)	1666
Figure 748. DSPI DSI Deserialization Data Register 0 (DSPIx_DDR0)	1667
Figure 749. DSPI DSI Configuration Register 1 (DSPIx_DSICR1)	1668
Figure 750. DSPI DSI Serialization Source Select Register 0 (DSPIx_SSR0)	1670
Figure 751. DSPI DSI Parallel Input Select Register 0 (DSPIx_PISR0)	1671
Figure 752. DSPI DSI Parallel Input Select Register 1 (DSPIx_PISR1)	1673
Figure 753. DSPI DSI Parallel Input Select Register 2 (DSPIx_PISR2)	1674
Figure 754. DSPI DSI Parallel Input Select Register 3 (DSPIx_PISR3)	1675
Figure 755. DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPIx_DIMR0)	1676
Figure 756. DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPIx_DPIR0)	1677
Figure 757. DSPI DSI Serialization Data Register 1 (DSPIx_SDR1)	1678
Figure 758. DSPI DSI Alternate Serialization Data Register 1 (DSPIx_ASDR1)	1679
Figure 759. DSPI DSI Transmit Comparison Register 1 (DSPIx_COMPR1)	1680
Figure 760. DSPI DSI Deserialization Data Register 1 (DSPIx_DDR1)	1681
Figure 761. DSPI DSI Serialization Source Select Register (DSPIx_SSR)	1682
Figure 762. DSPI DSI Parallel Input Select Register 4 (DSPIx_PISR4)	1683
Figure 763. DSPI DSI Parallel Input Select Register 5 (DSPIx_PISR5)	1685
Figure 764. DSPI DSI Parallel Input Select Register 6 (DSPIx_PISR6)	1686
Figure 765. DSPI DSI Parallel Input Select Register 7 (DSPIx_PISR7)	1687

Figure 766. DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPIx_DIMR1)	1688
Figure 767. DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPIx_DPIR1)	1689
Figure 768. DSPI Clock and Transfer Attributes Register Extended (DSPIx_CTAREn)	1690
Figure 769. DSPI Status Register Extended (DSPIx_SREX)	1691
Figure 770. SPI serial protocol overview	1693
Figure 771. DSI serialization diagram	1700
Figure 772. DSI deserialization diagram	1700
Figure 773. Example of parallel chaining of DSPIs	1702
Figure 774. Example of serial chaining of DSPIs.	1703
Figure 775. Example of System using DSPI in CSI Configuration.	1704
Figure 776. Communications clock prescalers and scalers.	1705
Figure 777. Peripheral Chip Select Strobe Timing.	1707
Figure 778. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)	1709
Figure 779. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8)	1710
Figure 780. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$)	1712
Figure 781. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$)	1712
Figure 782. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$)	1713
Figure 783. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$)	1713
Figure 784. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$)	1714
Figure 785. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$)	1714
Figure 786. Example of non-continuous format (CPHA = 1, CONT = 0)	1715
Figure 787. Example of continuous transfer (CPHA = 1, CONT = 1).	1715
Figure 788. Example of Fast Continuous Selection Format	1717
Figure 789. Continuous SCK timing diagram (CONT = 0).	1718
Figure 790. Continuous SCK Timing Diagram (CONT = 1).	1719
Figure 791. DSPI usage in the TSB Configuration	1720
Figure 792. TSB Downstream Frames	1720
Figure 793. TSB Data Frame Format for MSC Dual Receiver Operation	1722
Figure 794. Sample MSC downstream transmission using ITSB mode	1724
Figure 795. TX FIFO pointers and counter	1733
Figure 796. LFAST block diagram	1735
Figure 797. LFAST frame structure	1737
Figure 798. Serial frame structure	1738
Figure 799. Key to Register Fields.	1741
Figure 800. Mode Configuration Register (MCR)	1742
Figure 801. Speed Control Register (SCR)	1744
Figure 802. Correlator Control Register (COCR)	1745
Figure 803. Test Mode Control Register (TMCR)	1747
Figure 804. Auto Loopback Control Register (ALCR)	1748
Figure 805. Rate Change Delay Control Register (RCDCR)	1749
Figure 806. Wakeup Delay Control Register (SLCR)	1750
Figure 807. ICLC Control Register (ICR)	1752
Figure 808. Ping Control Register (PICR)	1753
Figure 809. Rx FIFO CTS Control Register (RFCR)	1754
Figure 810. Tx Interrupt Enable Register (TIER)	1755
Figure 811. Rx Interrupt Enable Register (RIER)	1756
Figure 812. Rx ICLC Interrupt Enable Register (RIIER)	1757
Figure 813. PLL Control Register (PLLCR)	1759
Figure 814. LVDS Control Register (LCR)	1761
Figure 815. Unsolicited Tx Control Register (UNSTCR)	1763
Figure 816. Unsolicited Tx Data Registers (UNSTDRO–UNSTD8)	1764
Figure 817. Global Status Register (GSR)	1764

Figure 818. Ping Status Register (PISR)	1766
Figure 819. Data frame Status Register (DFSR)	1766
Figure 820. Tx Interrupt Status Register (TISR)	1767
Figure 821. Rx Interrupt Status Register (RISR)	1768
Figure 822. Rx ICLC Interrupt Status Register (RIISR)	1770
Figure 823. PLL and LVDS Status Register (PLLSR)	1771
Figure 824. Unsolicited Rx Status Register (UNSRSR)	1772
Figure 825. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)	1773
Figure 826. Top Level Receive Controller	1777
Figure 827. Hunt Correlation mode	1778
Figure 828. High Speed 8 Phase Selection	1778
Figure 829. High Speed 8 Phase Clock Alignment Example	1779
Figure 830. High Speed 4 Phase Clock Alignment Example	1779
Figure 831. Low speed 4 phase clock alignment example	1780
Figure 832. Samplers, each Sampler has 3 Registers	1782
Figure 833. Sampler 5 Logic	1783
Figure 834. Extracting Header and Payload from the 16-bit output from Auto correlation.	1784
Figure 835. Transmit Controller Connections	1786
Figure 836. Example of Sleep mode	1788
Figure 837. CTS generation.	1790
Figure 838. Rx LoopBack mode	1797
Figure 839. Rx LVDS LoopBack mode	1798
Figure 840. Tx LoopBack mode without automatic frame generation	1799
Figure 841. Automatic Loopback Test status signal timings	1800
Figure 842. Tx LoopBack mode with automatic frame generation.	1801
Figure 843. Tx LVDS Loopback (external) mode with automatic frame generation.	1802
Figure 844. LFAST module clock domains	1807
Figure 845. External clock muxing of lfast_sysclk	1808
Figure 846. External clock muxing of lfast_sysclk of 20/26 MHz	1809
Figure 847. External clock muxing of lfast_sysclk of 10 MHz	1809
Figure 848. Slow speed 4 phases generation	1810
Figure 849. Clock Enables and Clock Paths	1810
Figure 850. Transmit Clocks Muxing	1812
Figure 851. FEC block diagram	1814
Figure 852. Interrupt Event Register (EIR)	1823
Figure 853. Ethernet Interrupt Mask Register (EIMR)	1824
Figure 854. Receive Descriptor Active Register (RDAR)	1826
Figure 855. Transmit Descriptor Active Register (TDAR)	1826
Figure 856. Ethernet Control Register (ECR)	1827
Figure 857. MII Management Frame Register (MMFR)	1828
Figure 858. MII Speed Control Register (MSCR)	1829
Figure 859. MIB Control Register (MIBC)	1830
Figure 860. Receive Control Register (RCR)	1831
Figure 861. Transmit Control Register (TCR)	1833
Figure 862. Physical Address Lower Register (PALR)	1834
Figure 863. Physical Address Upper Register (PAUR)	1834
Figure 864. Opcode/Pause Duration Register (OPD)	1835
Figure 865. Descriptor Individual Upper Address Register (IAUR)	1835
Figure 866. Descriptor Individual Lower Address Register (IALR)	1836
Figure 867. Descriptor Group Upper Address Register (GAUR)	1836
Figure 868. Descriptor Group Lower Address Register (GALR)	1837
Figure 869. Transmit FIFO Watermark Register (TFWR)	1837

Figure 870. FIFO Receive Bound Register (FRBR)	1838
Figure 871. FIFO Receive Start Register (FRSR)	1838
Figure 872. Receive Descriptor Ring Start Register (ERDSR)	1839
Figure 873. Transmit Buffer Descriptor Ring Start Register (ETDSR)	1839
Figure 874. Receive Buffer Size Register (EMRBR)	1840
Figure 875. Count of frames not counted correctly (RMON_T_DROP)	1840
Figure 876. RMON Tx packet count (RMON_T_PACKETS)	1840
Figure 877. RMON Tx broadcast packets (RMON_T_BC_PKT)	1841
Figure 878. RMON Tx multicast packets (RMON_T_MC_PKT)	1841
Figure 879. RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN)	1842
Figure 880. RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)	1842
Figure 881. RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)	1842
Figure 882. RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)	1843
Figure 883. RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB)	1843
Figure 884. RMON Tx collision count (RMON_T_COL)	1843
Figure 885. RMON Tx 64 byte packets (RMON_T_P64)	1844
Figure 886. RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)	1844
Figure 887. RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)	1844
Figure 888. RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)	1845
Figure 889. RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)	1845
Figure 890. RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047)	1845
Figure 891. RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)	1846
Figure 892. RMON Tx Octets (RMON_T_OCTETS)	1846
Figure 893. Count of transmitted frames not counted correctly (IEEE_T_DROP)	1846
Figure 894. Frames transmitted OK (IEEE_T_FRAME_OK)	1847
Figure 895. Frames transmitted with single collision (IEEE_T_1COL)	1847
Figure 896. Frames transmitted with multiple collisions (IEEE_T_MCOL)	1847
Figure 897. Frames transmitted after deferral delay (IEEE_T_DEF)	1848
Figure 898. Frames transmitted with late collision (IEEE_T_LCOL)	1848
Figure 899. Frames transmitted with excessive collisions (IEEE_T_EXCOL)	1848
Figure 900. Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR)	1849
Figure 901. Frames transmitted with carrier sense error (IEEE_T_CSERR)	1849
Figure 902. Frames transmitted with SQE error (IEEE_T_SQE)	1849
Figure 903. Flow control pause frames transmitted (IEEE_T_FDXFC)	1850
Figure 904. Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)	1850
Figure 905. Count of received frames not counted correctly (RMON_R_DROP)	1850
Figure 906. RMON Rx packet count (RMON_R_PACKETS)	1851
Figure 907. RMON Rx broadcast packets (RMON_R_BC_PKT)	1851
Figure 908. RMON Rx multicast packets (RMON_R_MC_PKT)	1852
Figure 909. RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN)	1852
Figure 910. RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)	1852
Figure 911. RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)	1853
Figure 912. RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)	1853
Figure 913. RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB)	1853
Figure 914. Reserved (RMON_R_RESVD_0)	1854
Figure 915. RMON Rx 64 byte packets (RMON_R_P64)	1854
Figure 916. RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)	1854
Figure 917. RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)	1855
Figure 918. RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)	1855
Figure 919. RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023)	1855
Figure 920. RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047)	1856
Figure 921. RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048)	1856

Figure 922. RMON Rx octets (RMON_R_OCTETS)	1856
Figure 923. Count of received frames not counted correctly (IEEE_R_DROP)	1857
Figure 924. Frames received OK (IEEE_R_FRAME_OK)	1857
Figure 925. Frames received with CRC error (IEEE_R_CRC)	1857
Figure 926. Frames received with alignment error (IEEE_R_ALIGN)	1858
Figure 927. Receive FIFO overflow count (IEEE_R_MACERR)	1858
Figure 928. Flow control pause frames received (IEEE_R_FDXFC)	1858
Figure 929. Octet count for frames received without error (IEEE_R_OCTETS_OK)	1859
Figure 930. MII data frame format	1859
Figure 931. MII nibble/octet to octet/nibble mapping	1860
Figure 932. MII management frame format	1860
Figure 933. Receive Buffer Descriptor (RxBD)	1863
Figure 934. Transmit Buffer Descriptor (TxBD)	1865
Figure 935. Ethernet address recognition—receive block decisions	1872
Figure 936. Ethernet address recognition—microcode decisions	1873
Figure 937. RMII loopback mode	1877
Figure 938. RMII echo mode	1878
Figure 939. FlexRay block diagram	1882
Figure 940. Module Version Register (FR_MVR)	1897
Figure 941. Module Configuration Register (FR_MCR)	1897
Figure 942. System Memory Base Address High Register (FR_SYMBADHR)	1900
Figure 943. System Memory Base Address Low Register (FR_SYMBADLR)	1900
Figure 944. Strobe Signal Control Register (FR_STBSCR)	1900
Figure 945. Message Buffer Data Size Register (FR_MBDSR)	1902
Figure 946. Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)	1902
Figure 947. PE DRAM Access Register (FR_PEDRAR)	1903
Figure 948. PE DRAM Data Register (FR_PEDRDR)	1904
Figure 949. Protocol Operation Control Register (FR_POCR)	1905
Figure 950. Global Interrupt Flag and Enable Register (FR_GIFER)	1906
Figure 951. Protocol Interrupt Flag Register 0 (FR_PIFR0)	1909
Figure 952. Protocol Interrupt Flag Register 1 (FR_PIFR1)	1911
Figure 953. Protocol Interrupt Enable Register 0 (FR_PIER0)	1912
Figure 954. Protocol Interrupt Enable Register 1 (FR_PIER1)	1914
Figure 955. CHI Error Flag Register (FR_CHIERFR)	1915
Figure 956. Message Buffer Interrupt Vector Register (FR_MBIVEC)	1918
Figure 957. Channel A Status Error Counter Register (FR_CASERCR)	1918
Figure 958. Channel B Status Error Counter Register (FR_CBSERCR)	1919
Figure 959. Protocol Status Register 0 (FR_PSR0)	1919
Figure 960. Protocol Status Register 1 (FR_PSR1)	1921
Figure 961. Protocol Status Register 2 (FR_PSR2)	1922
Figure 962. Protocol Status Register 3 (FR_PSR3)	1924
Figure 963. Macrotick Counter Register (FR_MTCTR)	1925
Figure 964. Cycle Counter Register (FR_CYCTR)	1926
Figure 965. Slot Counter Channel A Register (FR_SLTCTAR)	1926
Figure 966. Slot Counter Channel B Register (FR_SLTCTBR)	1927
Figure 967. Rate Correction Value Register (FR_RTCORVR)	1927
Figure 968. Offset Correction Value Register (FR_OFCORVR)	1928
Figure 969. Combined Interrupt Flag Register (FR_CIFR)	1928
Figure 970. System Memory Access Time-Out Register (FR_SYMATOR)	1929
Figure 971. Sync Frame Counter Register (FR_SFCNTR)	1930
Figure 972. Sync Frame Table Offset Register (FR_SFTOR)	1931
Figure 973. Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)	1931

Figure 974. Sync Frame ID Rejection Filter Register (FR_SFIDRFR)	1933
Figure 975. Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)	1933
Figure 976. Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)	1934
Figure 977. Network Management Vector Registers (FR_NMVR0–FR_NMVR5)	1934
Figure 978. Network Management Vector Length Register (FR_NMVLR)	1935
Figure 979. Timer Configuration and Control Register (FR_TICCR)	1936
Figure 980. Timer 1 Cycle Set Register (FR_TI1CYSR)	1937
Figure 981. Timer 1 Macro-tick Offset Register (FR_TI1MTOR)	1937
Figure 982. Timer 2 Configuration Register 0 (FR_TI2CR0)	1938
Figure 983. Timer 2 Configuration Register 1 (FR_TI2CR1)	1938
Figure 984. Slot Status Selection Register (FR_SSSR)	1939
Figure 985. Slot Status Counter Condition Register (FR_SSCCR)	1940
Figure 986. Slot Status Registers (FR_SSR0–FR_SSR7)	1942
Figure 987. Slot Status Counter Registers (FR_SSCR0–FR_SSCR3)	1944
Figure 988. MTS A Configuration Register (FR_MTSACFR)	1944
Figure 989. MTS B Configuration Register (FR_MTSBCFR)	1945
Figure 990. Receive Shadow Buffer Index Register (FR_RSBR)	1945
Figure 991. Receive FIFO Start Data Offset Register (FR_RFSODR)	1946
Figure 992. Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)	1947
Figure 993. Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)	1947
Figure 994. Receive FIFO Periodic Timer Register (FR_RFPTTR)	1947
Figure 995. Receive FIFO Watermark and Selection Register (FR_RFWMSR)	1948
Figure 996. Receive FIFO Start Index Register (FR_RFSIR)	1949
Figure 997. Receive FIFO Depth and Size Register (FR_RFDSR)	1949
Figure 998. Receive FIFO A Read Index Register (FR_RFARIR)	1950
Figure 999. Receive FIFO B Read Index Register (FR_RFBIR)	1950
Figure 1000. Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)	1951
Figure 1001. Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)	1951
Figure 1002. Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)	1952
Figure 1003. Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)	1952
Figure 1004. Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)	1953
Figure 1005. Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)	1953
Figure 1006. Receive FIFO Range Filter Control Register (FR_RFRFCTR)	1954
Figure 1007. Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)	1955
Figure 1008. Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)	1955
Figure 1009. Protocol Configuration Register 0 (FR_PCR0)	1958
Figure 1010. Protocol Configuration Register 1 (FR_PCR1)	1958
Figure 1011. Protocol Configuration Register 2 (FR_PCR2)	1959
Figure 1012. Protocol Configuration Register 3 (FR_PCR3)	1959
Figure 1013. Protocol Configuration Register 4 (FR_PCR4)	1959
Figure 1014. Protocol Configuration Register 5 (FR_PCR5)	1959
Figure 1015. Protocol Configuration Register 6 (FR_PCR6)	1960
Figure 1016. Protocol Configuration Register 7 (FR_PCR7)	1960
Figure 1017. Protocol Configuration Register 8 (FR_PCR8)	1960
Figure 1018. Protocol Configuration Register 9 (FR_PCR9)	1960
Figure 1019. Protocol Configuration Register 10 (FR_PCR10)	1961
Figure 1020. Protocol Configuration Register 11 (FR_PCR11)	1961
Figure 1021. Protocol Configuration Register 12 (FR_PCR12)	1961
Figure 1022. Protocol Configuration Register 13 (FR_PCR13)	1961
Figure 1023. Protocol Configuration Register 14 (FR_PCR14)	1962
Figure 1024. Protocol Configuration Register 15 (FR_PCR15)	1962
Figure 1025. Protocol Configuration Register 16 (FR_PCR16)	1962

Figure 1026. Protocol Configuration Register 17 (FR_PCR17)	1962
Figure 1027. Protocol Configuration Register 18 (FR_PCR18)	1963
Figure 1028. Protocol Configuration Register 19 (FR_PCR19)	1963
Figure 1029. Protocol Configuration Register 20 (FR_PCR20)	1963
Figure 1030. Protocol Configuration Register 21 (FR_PCR21)	1963
Figure 1031. Protocol Configuration Register 22 (FR_PCR22)	1964
Figure 1032. Protocol Configuration Register 23 (FR_PCR23)	1964
Figure 1033. Protocol Configuration Register 24 (FR_PCR24)	1964
Figure 1034. Protocol Configuration Register 25 (FR_PCR25)	1964
Figure 1035. Protocol Configuration Register 26 (FR_PCR26)	1965
Figure 1036. Protocol Configuration Register 27 (FR_PCR27)	1965
Figure 1037. Protocol Configuration Register 28 (FR_PCR28)	1965
Figure 1038. Protocol Configuration Register 29 (FR_PCR29)	1965
Figure 1039. Protocol Configuration Register 30 (FR_PCR30)	1966
Figure 1040. ECC Error Interrupt Flag and Enable Register (FR_EEIFER)	1966
Figure 1041. ECC Error Report and Injection Control Register (FR_EERICR)	1969
Figure 1042. ECC Error Report Address Register (FR_EERAR)	1970
Figure 1043. ECC Error Report Data Register (FR_EERDR)	1971
Figure 1044. ECC Error Report Code Register (FR_EERCR)	1972
Figure 1045. ECC Error Injection Address Register (FR_EEIAR)	1972
Figure 1046. ECC Error Injection Data Register (FR_EEIDR)	1973
Figure 1047. ECC Error Injection Code Register (FR_EEICR)	1974
Figure 1048. Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)	1974
Figure 1049. Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)	1976
Figure 1050. Message Buffer Frame ID Registers (FR_MBFIDRn)	1977
Figure 1051. Message Buffer Index Registers (FR_MBIDXRn)	1977
Figure 1052. Message Buffer Data Field Offset Registers (FR_MBDORn)	1978
Figure 1053. LRAM ECC Error Test Registers (FR_LEETRn)	1978
Figure 1054. Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)	1979
Figure 1055. Stop Watch Count High Register (FR_STPWHR)	1979
Figure 1056. Stop Watch Count Low Register (FR_STPWLRL)	1980
Figure 1057. Physical message buffer structure	1981
Figure 1058. Individual message buffer structure	1983
Figure 1059. Receive shadow buffer structure	1984
Figure 1060. Receive FIFO structure	1986
Figure 1061. Example of FlexRay memory area layout (FR_MCR[FAM] = 0)	1989
Figure 1062. AreaExample of FlexRay memory area layout (FR_MCR[FAM] = 1)	1990
Figure 1063. Frame header structure (receive message buffer and receive FIFO)	1992
Figure 1064. Frame header structure (transmit message buffer)	1992
Figure 1065. Frame header structure (transmit message buffer for key slot)	1993
Figure 1066. Receive message buffer slot status structure (ChAB)	1995
Figure 1067. Receive message buffer slot status structure (ChA)	1995
Figure 1068. Receive message buffer slot status structure (ChB)	1996
Figure 1069. Transmit message buffer slot status structure (ChAB)	1997
Figure 1070. Transmit message buffer slot status structure (ChA)	1997
Figure 1071. Transmit message buffer slot status structure (ChB)	1998
Figure 1072. Message buffer data field structure	1999
Figure 1073. Transmit message buffer access regions	2003
Figure 1074. Transmit message buffer states	2004
Figure 1075. Message transmission timing	2008
Figure 1076. Message transmission from HLck state with unlock	2008
Figure 1077. Null frame transmission from Idle state	2009

Figure 1078. Null frame transmission from HLck state	2009
Figure 1079. Null frame transmission from HLck state with unlock	2009
Figure 1080. Null frame transmission from Idle state with locking	2009
Figure 1081. Receive message buffer access regions	2011
Figure 1082. Receive message buffer states	2012
Figure 1083. Message Reception timing	2016
Figure 1084. Inconsistent channel assignment	2019
Figure 1085. Message buffer reconfiguration scheme	2021
Figure 1086. Received frame FIFO filter path	2025
Figure 1087. Dual-Channel Device mode	2028
Figure 1088. Single-Channel Device mode (channel A)	2029
Figure 1089. Single-Channel Device mode (channel B)	2029
Figure 1090. External offset correction write and application timing	2030
Figure 1091. External rate correction write and application timing	2030
Figure 1092. Sync table memory layout	2031
Figure 1093. Sync frame table trigger and generation timing	2033
Figure 1094. Strobe signal timing (type = pulse, clk_offset = -2)	2036
Figure 1095. Strobe signal timing (type = pulse, clk_offset = +4)	2036
Figure 1096. Slot status vector update	2038
Figure 1097. Slot status counting and FR_SSCRn update	2040
Figure 1098. Scheme of FR_GIFER interrupt signal generation	2044
Figure 1099. FR_EEIFER interrupt signal generation scheme	2045
Figure 1100. FR_CIFR flag generation scheme	2046
Figure 1101. StopWatch counter logic	2056
Figure 1102. StopWatch function timing diagram	2056
Figure 1103. Transmit data not available	2064
Figure 1104. Transmit data available	2064
Figure 1105. I ² C block diagram	2066
Figure 1106. START and STOP conditions	2068
Figure 1107. Register figure conventions	2070
Figure 1108. I ² C Bus Address register (IBAD)	2071
Figure 1109. I ² C Bus Frequency Divider Register (IBFD)	2071
Figure 1110. I ² C Bus Control Register (IBCR)	2072
Figure 1111. I ² C Bus Status Register (IBSR)	2073
Figure 1112. I ² C Bus Data I/O Register (IBDR)	2075
Figure 1113. I ² C Bus Interrupt Config Register (IBIC)	2076
Figure 1114. I ² C Bus Debug Register (IBDBG)	2076
Figure 1115. I ² C transaction protocol	2078
Figure 1116. I ² C bus clock synchronization	2081
Figure 1117. SDA Hold time	2083
Figure 1118. SCL Divider and SDA Hold	2083
Figure 1119. IPG DEBUG mode	2091
Figure 1120. IPG debug mode with repeated start	2092
Figure 1121. Recommended I ² C interrupt service routine flowchart	2094
Figure 1122. Flow-Chart of DMA mode master transmit	2100
Figure 1123. Flow-Chart of DMA mode master receive	2101
Figure 1124. PSI5 digital interface block diagram	2105
Figure 1125. Block Diagram of Debug Mode	2110
Figure 1126. Operating modes	2111
Figure 1127. Global Control Register (PSI5_GCR)	2136
Figure 1128. PSI5 Channel Control Register (PSI5_PCCR)	2137
Figure 1129. DMA Control Register (PSI5_DCR)	2143

Figure 1130. DMA Status Register (PSI5_DSR)	2146
Figure 1131. General Interrupt Control Register (PSI5_GICR)	2148
Figure 1132. New Data Interrupt Control Register (PSI5_NDICR)	2149
Figure 1133. Overwrite Interrupt Control Register (PSI5_OWICR)	2150
Figure 1134. Error Interrupt Control Register (PSI5_EICR)	2151
Figure 1135. General Interrupt Status Register (PSI5_GISR)	2152
Figure 1136. DMA PSI5 Message Register (PSI5_DPMR)	2154
Figure 1137. DMA SMC Frame Register (PSI5_DSFR)	2155
Figure 1138. DMA Diagnostic Status Register (PSI5_DDSR)	2155
Figure 1139. PSI5 Message Receive Register Low (PSI5_PMRRL)	2156
Figure 1140. PSI5 Message Receive Register High (PSI5_PMRRH)	2157
Figure 1141. PSI5 Message Register (Lower Word) {i:0 to 31} (PSI5_PMRLn)	2158
Figure 1142. PSI5 Message Register High <i>n</i> (PSI5_PMRHn)	2159
Figure 1143. SMC Frame Register <i>n</i> (PSI5_SFRn)	2161
Figure 1144. New Data Status Register (PSI5_NDSR)	2163
Figure 1145. Overwrite Status Register (OWSR)	2163
Figure 1146. Error Indication Status Register (PSI5_EISR)	2164
Figure 1147. Set New Data Status Register (PSI5_SNDSR)	2165
Figure 1148. Set Overwrite Status Register (PSI5_SOWSR)	2165
Figure 1149. Set Error Status Register (PSI5_SEISR)	2166
Figure 1150. Set SMC Error Status Register (PSI5_SSESR)	2167
Figure 1151. Sync Time Stamp Read Register (PSI5_STSR)	2168
Figure 1152. Data Time Stamp Read Register (DTSRR)	2168
Figure 1153. Slot <i>n</i> Frame Configuration Register (PSI5_SnFCR)	2169
Figure 1154. Slot <i>n</i> Start Boundary Register (PSI5_SnSBR)	2170
Figure 1155. Slot <i>n</i> End Boundary Register (PSI5_SnEBR)	2171
Figure 1156. Data Output Block Configuration Register (PSI5_DOBCR)	2172
Figure 1157. Manchester Decoder Disable Offset Register (PSI5_MDDIS_OFF)	2176
Figure 1158. PSI5_Pulse Width for Data Bit 0 Register (PW0D)	2176
Figure 1159. PSI5_Pulse Width for Data Bit 1 Register (PW1D)	2177
Figure 1160. Counter Target Pulse Register (PSI5_CTPR)	2177
Figure 1161. Counter Initialize Pulse Register (PSI5_CIPR)	2178
Figure 1162. Data Preparation Register Low (PSI5_DPRL)	2179
Figure 1163. Data Preparation Register High (PSI5_DPRH)	2180
Figure 1164. Data Buffer Register Low (PSI5_DBR)	2181
Figure 1165. Data Buffer Register High (PSI5_DBRH)	2185
Figure 1166. Data Shift Register Low (PSI5_DSRL)	2187
Figure 1167. Data Shift Register High (PSI5_DSRH)	2188
Figure 1168. System setup (peripheral sensor connected to device)	2190
Figure 1169. Bit encoding of data from sensor	2191
Figure 1170. Receive data frame format	2193
Figure 1171. Data frame configuration and sampling	2194
Figure 1172. Start Bit Detection	2195
Figure 1173. Bit Extraction	2196
Figure 1174. Error Free Case	2198
Figure 1175. Message Too Early	2199
Figure 1176. Message Too Late	2200
Figure 1177. Message Missing	2201
Figure 1178. Message Much Too Late (in unconfigured slot)	2202
Figure 1179. Message Much Too Late (in unconfigured slot)	2203
Figure 1180. Message Too Late (in last slot)	2204
Figure 1181. Message Too Late (in last slot)	2205

Figure 1182. Message Too Early (in first slot)	2206
Figure 1183. Message Too Early (in first slot)	2207
Figure 1184. Message Too Late (overlaps next message)	2208
Figure 1185. Diagnosis Information attached to every frame	2209
Figure 1186. PSI5 and SMC Message Read Logic Block	2211
Figure 1187. CRC calculation scheme	2211
Figure 1188. Standard Frame Formats Supported	2213
Figure 1189. Data Transmission Block	2216
Figure 1190. ECU-to-sensor communication registers (implementation of Figure 1189)	2221
Figure 1191. Synchronous mode communication scenarios	2222
Figure 1192. PSI5 clock scheme	2228
Figure 1193. Fast initialization phase	2230
Figure 1194. CIPR and CTPR	2232
Figure 1195. Sync Pulse Generation Parameters and Jitters	2233
Figure 1196. ISPG (Internal Sync Pulse Generator) Structure	2234
Figure 1197. ECU to Sensor Communication States	2235
Figure 1198. Implementation of EI Diagnostic Register	2236
Figure 1199. PSI5-S digital interface block diagram	2239
Figure 1200. Operating modes	2242
Figure 1201. PSI5-S LIN Control Register 1 (PS_LINCR1)	2255
Figure 1202. PSI5-S LIN Interrupt enable register (PS_LINIER)	2257
Figure 1203. LIN Status Register (LINSR)	2259
Figure 1204. PSI5-S UART Mode Control Register (PS_UARTCR)	2259
Figure 1205. PSI5-S UART Mode Status Register (PS_UARTSR)	2262
Figure 1206. PSI5-S LIN Fractional Baud Rate Register (PS_LINFBRR)	2264
Figure 1207. PSI5-S LIN Integer Baud Rate Register (PS_LINIBRR)	2265
Figure 1208. PSI5-S LIN Control Register 2 (PS_LINCR2)	2267
Figure 1209. PSI5-S Buffer Data Register Least Significant (PS_BDRL)	2268
Figure 1210. PSI5-S Buffer Data Register Most Significant (PS_BDRM)	2269
Figure 1211. PSI5-S Global Control Register (PS_GCR)	2270
Figure 1212. PSI5-S UART Preset Timeout Register (PS_UARTPTO)	2271
Figure 1213. PSI5-S UART Current Timeout Register (PS_UARTCTO)	2272
Figure 1214. DMA Tx Enable Register (DMATXE)	2273
Figure 1215. DMA Rx Enable Register (DMARXE)	2274
Figure 1216. PSI5-S Global Control Register (PS_GLCR)	2275
Figure 1217. PSI5-S Global Status Register (PS_GLSR)	2277
Figure 1218. PSI5-S Channel Base Address Register (PS_CH_BASE_ADDR)	2278
Figure 1219. PSI5-S MRU Output Buffer2 Register0 (PS_MRU_BUF2_REG0)	2279
Figure 1220. PSI5-S MRU OUTPUT BUFFER2 REGISTER1 (PS_MRU_BUF2_REG1)	2280
Figure 1221. PSI5-S MRU OUTPUT BUFFER REGISTER2 (PS_MRU_BUF2_REG2)	2282
Figure 1222. PSI5-S MRU OUTPUT BUFFER2 REGISTER3 (PS_MRU_BUF2_REG3)	2282
Figure 1223. PSI5-S Clock Divider "0" for WD and DDSR trigger counters (PS_WD_DDTRIG_CLKDIV0)	2283
Figure 1224. PSI5-S Clock Divider "1" for WD and DDSR trigger counters (PS_WD_DDTRIG_CLKDIV1)	2284
Figure 1225. PSI5-S Clock Divider "2" for WD and DDSR trigger counters (PS_WD_DDTRIG_CLKDIV2)	2285
Figure 1226. PSI5-S Clock Divider "3" for WD and DDSR trigger counters (PS_WD_DDTRIG_CLKDIV3)	2286
Figure 1227. PSI5-S Mbox Status Irq (PS_MBOX_SR_IRQ)	2287
Figure 1228. PSI5-S Error Status Irq (PS_ERR_SR_IRQ)	2288
Figure 1229. PSI5-S Mailbox select IRQ[irq_n](PS_MBOX_SEL_IRQ[irq_n])	2293

Figure 1230. PSi5-S Error Enable irq[irq_n] (PS_ERR_SEL_IRQ[irq_n])	2294
Figure 1231. PSi5-S channel 0 message configuration register A (PS_WDGTSSR)	2296
Figure 1232. PSi5-S channel 0 message configuration register A (PS_MSGA_CH0)	2297
Figure 1233. PSi5-S channel 0 message configuration register B (PS_MSGB_CH0)	2298
Figure 1234. PSi5-S Channel 0 Mailbox ID (PS_MBOX_SR_CH0)	2299
Figure 1235. PSi5-S channel message configuration register A (PS_MSGA_CHn)	2301
Figure 1236. PSi5-S channel message configuration register B (PS_MSGB_CHn)	2303
Figure 1237. PSi5-S Channel Mailbox ID (PS_MBOX_SR_CHn)	2305
Figure 1238. PSi5-S channel watchdog configuration register (PS_WD_CFGR_CHn)	2310
Figure 1239. PSi5-S DDSR Trigger offset register channel (PS_DDTRIG_OFFR_CHn)	2311
Figure 1240. PSi5-S DDSR Trigger period register channel (PS_DDTRIG_PERR_CHn)	2312
Figure 1241. PSi5-S ECU to Sensor Control Register (PS_E2SCR_CHn)	2313
Figure 1242. PSi5-S ECU to Sensor Status Register (PS_E2SSR_CHn)	2316
Figure 1243. PSi5-S channel1 ECU to Sensor Upstream Data Shift Register High (PS_DDSCR_H_CHn)	2317
Figure 1244. PSi5-S channel1 ECU to Sensor Upstream Data Shift Register Low (PS_DDSCR_L_CHn)	2318
Figure 1245. System setup (peripheral sensor connected to device)	2319
Figure 1246. PSi5-S Packet Structure	2319
Figure 1247. PSi5-S message handling and Error Flags	2323
Figure 1248. MRU buf1 structure	2324
Figure 1249. MRU buf2 structure	2324
Figure 1250. CRC calculation scheme	2326
Figure 1251. XCRC calculation	2327
Figure 1252. Standard Frame Formats Supported	2329
Figure 1253. DDSR internal shift trigger generator	2332
Figure 1254. PS_DDSCR Sensor Command (Ex: XXL Long Frame)	2333
Figure 1255. PS_DDSCR Sensor Command (Ex: PSi5 LONG format)	2334
Figure 1256. CMD and ACMD	2334
Figure 1257. TimeStamp Capture	2336
Figure 1258. Watchdog Counter	2337
Figure 1259. IRQ[n] interrupts	2339
Figure 1260. ECU to Sensor interrupts	2340
Figure 1261. Global Interrupts	2340
Figure 1262. Clock distribution in PSi5-S	2341
Figure 1263. SENT block diagram	2345
Figure 1264. Global Control Register (GBL_CTRL)	2350
Figure 1265. Channel Enable Register (CHNL_EN)	2352
Figure 1266. Global Status Register (GBL_STATUS)	2353
Figure 1267. Fast Message Ready Status Register (FMSG_RDY)	2355
Figure 1268. Slow Serial Message Ready Status Register (SMSG_RDY)	2357
Figure 1269. Data Control Register 1 (DATA_CTRL1)	2359
Figure 1270. Data Control Register 2 (DATA_CTRL2)	2361
Figure 1271. Fast Message DMA Control Register (FDMA_CTRL)	2363
Figure 1272. Slow Serial Message DMA Control Register (SDMA_CTRL)	2364
Figure 1273. Fast Message Ready Interrupt Enable Register (FRDY_IE)	2365
Figure 1274. Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)	2366
Figure 1275. DMA Fast Message Data Read Register (DMA_FMSG_DATA)	2367
Figure 1276. DMA Fast Message CRC Read Register (DMA_FMSG_CRC)	2368
Figure 1277. DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)	2369
Figure 1278. DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)	2370
Figure 1279. DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)	2372

Figure 1280. DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)	2373
Figure 1281. Channel 'n' Clock Control Register (CHn_CLK_CTRL)	2374
Figure 1282. Channel 'n' Status Register (CHn_STATUS)	2376
Figure 1283. Channel 'n' Configuration Register (CHn_CONFIG)	2379
Figure 1284. Fast Message DMA read logic	2386
Figure 1285. Slow Serial Message DMA read logic	2388
Figure 1286. Enhanced Serial Message with 8-bit ID field	2388
Figure 1287. Enhanced Serial Message with 4-bit ID field	2388
Figure 1288. Short serial message	2389
Figure 1289. Input programmable filter timing diagram	2392
Figure 1290. SENT Receiver module clock domains	2396
Figure 1291. Block diagram	2399
Figure 1292. Frames	2402
Figure 1293. Structure of LIN frame	2402
Figure 1294. Structure of byte field	2402
Figure 1295. Break field	2403
Figure 1296. Sync byte field	2403
Figure 1297. Identifier field	2403
Figure 1298. Operating modes	2404
Figure 1299. Incomplete response (for example, missing checksum)	2409
Figure 1300. No response	2410
Figure 1301. Identifier List mode	2413
Figure 1302. Identifier Mask mode	2413
Figure 1303. Start detection and sampling for over sampling rate = 4 (Case 1)	2416
Figure 1304. Start detection and sampling for over sampling rate = 4 (Case 2a)	2416
Figure 1305. Start detection and sampling for over sampling rate = 4 (Case 2b)	2417
Figure 1306. Start detection and sampling for over sampling rate = 4 (Case 3)	2417
Figure 1307. Start detection and sampling for over sampling rate = 4 (Case 4)	2417
Figure 1308. Start detection and sampling for over sampling rate = 4 (Case 5)	2418
Figure 1309. Wakeup sequence	2419
Figure 1310. UART mode 8-bit data frame	2420
Figure 1311. UART mode 9-bit data frame	2420
Figure 1312. UART mode 16-bit data frame	2421
Figure 1313. UART mode 17-bit data frame	2421
Figure 1314. UART mode 13-bit data frame	2421
Figure 1315. Structure of 8-byte buffer	2422
Figure 1316. Master node — TX memory map	2428
Figure 1317. Master node — DMA Tx FSM (concept scheme)	2430
Figure 1318. Master node — RX memory map	2431
Figure 1319. Master node — DMA Rx FSM (concept scheme)	2432
Figure 1320. Slave node — TX memory map	2433
Figure 1321. Slave node — DMA Tx FSM (concept scheme)	2434
Figure 1322. Slave node — RX memory map	2435
Figure 1323. Slave node — DMA Rx FSM (concept scheme)	2437
Figure 1324. UART — TX memory map	2437
Figure 1325. UART — DMA Tx FSM (concept scheme)	2439
Figure 1326. UART — RX memory map	2439
Figure 1327. UART — DMA Rx FSM (concept scheme)	2441
Figure 1328. DMA interface (major loop — single iteration)	2442
Figure 1329. DMA interface (major loop — N iteration)	2442
Figure 1330. LIN Control Register 1 (LINCR1)	2445
Figure 1331. LIN Interrupt enable register (LINIER)	2448

Figure 1332. LIN Status Register (LINSR)	2450
Figure 1333. LIN error status register (LINESR)	2453
Figure 1334. UART Mode Control Register (UARTCR)	2455
Figure 1335. UART Mode Status Register (UARTSR)	2460
Figure 1336. LIN Time-Out Control Status Register (LINTCSR)	2463
Figure 1337. LIN Output Compare Register (LINOOCR)	2464
Figure 1338. LIN Time-Out Control Register (LINTOCR)	2464
Figure 1339. LIN Fractional Baud Rate Register (LINFBR)	2465
Figure 1340. LIN Integer Baud Rate Register (LINIBRR)	2466
Figure 1341. LIN Checksum Field Register (LINCFR)	2467
Figure 1342. LIN Control Register 2 (LINC2)	2468
Figure 1343. Buffer Identifier Register (BIDR)	2470
Figure 1344. Buffer Data Register Least Significant (BDRL)	2471
Figure 1345. Buffer Data Register Most Significant (BDRM)	2472
Figure 1346. Identifier Filter Enable Register (IFER)	2473
Figure 1347. Identifier Filter Match Index (IFMI)	2474
Figure 1348. Identifier Filter Mode Register (IFMR)	2475
Figure 1349. Identifier Filter Control Register (IFCR2n)	2476
Figure 1350. Identifier Filter Control Register (IFCR2n+1)	2477
Figure 1351. Global Control Register (GCR)	2478
Figure 1352. UART Preset Timeout Register (UARTPTO)	2480
Figure 1353. UART Current Timeout Register (UARTCTO)	2481
Figure 1354. DMA Tx Enable Register (DMATXE)	2482
Figure 1355. DMA Rx Enable Register (DMARXE)	2483
Figure 1356. Master node — transmitter configuration	2487
Figure 1357. Master node — receiver configuration	2487
Figure 1358. Master node — transmitter, bit error configuration	2487
Figure 1359. Master node — transmitter, checksum error configuration	2488
Figure 1360. Master node — receiver, checksum error configuration	2488
Figure 1361. Slave node — transmitter (no identifier filters) configuration	2488
Figure 1362. Slave node — receiver (no identifier filters) configuration	2489
Figure 1363. Slave node — receiver (no identifier filters) configuration	2489
Figure 1364. Slave node — No filters, transmitter, bit error configuration	2489
Figure 1365. Slave node — No filters, receiver, checksum error configuration	2489
Figure 1366. Slave node — one TX filter configuration	2490
Figure 1367. Slave node — one RX filter configuration	2490
Figure 1368. RX only, TX only, RX & TX filters configuration	2490
Figure 1369. TX filter, BF is set configuration — ID has matched	2491
Figure 1370. TX filter, BF is set configuration — ID not matched	2491
Figure 1371. RX filter, BF is set configuration — ID matched	2491
Figure 1372. RX filter, BF is set configuration — ID not matched (ID is Rx)	2491
Figure 1373. RX filter, BF is set configuration — (ID is Tx)	2492
Figure 1374. TX filter, RX filter, BF set configuration — (ID matched)	2492
Figure 1375. TX filter, RX filter, BF set configuration — (ID matched + 1)	2492
Figure 1376. TX filter, RX filter, BF set configuration — (ID not matched, RX/TX Interrupt)	2492
Figure 1377. Extended frames (TX Interrupt)	2493
Figure 1378. Extended frames (RX Interrupt)	2493
Figure 1379. Response timeout	2493
Figure 1380. Frame timeout	2494
Figure 1381. Header timeout	2494
Figure 1382. UART mode configuration	2494
Figure 1383. Interrupt diagram	2496

Figure 1384. MC_RGM block diagram	2501
Figure 1385. 'Destructive' Event Status Register (RGM_DES)	2515
Figure 1386. 'Destructive' Event Reset Disable Register (RGM_DERD)	2517
Figure 1387. 'Destructive' Event Alternate Request Register (RGM_DEAR)	2518
Figure 1388. 'Destructive' Bidirectional Reset Enable Register (RGM_DBRE)	2519
Figure 1389. 'Functional' Event Status Register (RGM_FES)	2521
Figure 1390. 'Functional' Event Reset Disable Register (RGM_FERD)	2523
Figure 1391. 'Functional' Event Alternate Request Register (RGM_FEAR)	2524
Figure 1392. 'Functional' Bidirectional Reset Enable Register (RGM_FBRE)	2525
Figure 1393. 'Functional' Event Short Sequence Register (RGM_FESS)	2527
Figure 1394. 'Functional' Reset Escalation Threshold Register (RGM_FRET)	2528
Figure 1395. 'Destructive' Reset Escalation Threshold Register (RGM_DRET)	2529
Figure 1396. External Reset Output Extension Control Register (RGM_EROEC)	2530
Figure 1397. Peripheral Reset Register 0 (RGM_PRST0)	2530
Figure 1398. Peripheral Reset Register 1 (RGM_PRST1)	2531
Figure 1399. Peripheral Reset Register 2 (RGM_PRST2)	2532
Figure 1400. Peripheral Reset Register 3 (RGM_PRST3)	2533
Figure 1401. Peripheral Reset Register 4 (RGM_PRST4)	2533
Figure 1402. Peripheral Reset Register 5 (RGM_PRST5)	2534
Figure 1403. Peripheral Reset Register 6 (RGM_PRST6)	2534
Figure 1404. Peripheral Reset Register 7 (RGM_PRST7)	2535
Figure 1405. Peripheral Reset Register 0 (RGM_PSTAT0)	2536
Figure 1406. Peripheral Reset Register 1 (RGM_PSTAT1)	2537
Figure 1407. Peripheral Reset Register 2 (RGM_PSTAT2)	2538
Figure 1408. Peripheral Reset Register 3 (RGM_PSTAT3)	2539
Figure 1409. Peripheral Reset Register 4 (RGM_PSTAT4)	2539
Figure 1410. Peripheral Reset Register 5 (RGM_PSTAT5)	2540
Figure 1411. Peripheral Reset Register 6 (RGM_PSTAT6)	2541
Figure 1412. Peripheral Reset Register 7 (RGM_PSTAT7)	2541
Figure 1413. MC_RGM state machine	2543
Figure 1414. Flow of control	2552
Figure 1415. SEC_SWT_CPU bit configuration	2557
Figure 1416. BAF DCF ADDR client configuration	2557
Figure 1417. DATA32 DCF client configuration	2558
Figure 1418. DATA16 DCF Client Configuration	2558
Figure 1419. DATA8 DCF Client Configuration	2559
Figure 1420. Boot header configuration	2561
Figure 1421. Production Disable callback function flow chart	2563
Figure 1422. Flow of control during serial boot	2564
Figure 1423. Start address	2567
Figure 1424. Download VLE bit and code size	2567
Figure 1425. LINFlexD bit timing in UART mode	2568
Figure 1426. CAN Bit Timing	2569
Figure 1427. SSCM block diagram	2572
Figure 1428. System Status Register (STATUS)	2574
Figure 1429. System memory and ID register (MEMCONFIG)	2575
Figure 1430. Error Configuration Register (ERROR)	2575
Figure 1431. HSM and User Option Status Register (UOPS)	2576
Figure 1432. Processor Start Address Register (PSA)	2577
Figure 1433. HSM Start Address Register (HSA)	2578
Figure 1434. Life Cycle Status Register (LCSTAT)	2578
Figure 1435. Appending DCF records	2581

Figure 1436. Digital PMC block diagram	2586
Figure 1437. Supply Gauge Register (<i>GR_S</i>)	2589
Figure 1438. Pending Gauge Status Register (<i>GR_P</i>)	2591
Figure 1439. Interrupt Enable Pending Register (<i>IE_P</i>)	2595
Figure 1440. Event Pending Register (<i>EPR_VD3</i>)	2598
Figure 1441. Reset Event Enable Register (<i>REE_VD3</i>)	2599
Figure 1442. Reset Event Select Register (<i>RES_VD3</i>)	2600
Figure 1443. FCCU Event Enable Register (<i>FEE_VD3</i>)	2601
Figure 1444. Event Pending Register (<i>EPR_VD4</i>)	2602
Figure 1445. Reset Event Enable Register (<i>REE_VD4</i>)	2603
Figure 1446. Reset Event Select Register (<i>RES_VD4</i>)	2604
Figure 1447. FCCU Event Enable Register (<i>FEE_VD4</i>)	2605
Figure 1448. Event Pending Register (<i>EPR_VD7</i>)	2605
Figure 1449. Reset Event Enable Register (<i>REE_VD7</i>)	2606
Figure 1450. Reset Event Select Register (<i>RES_VD7</i>)	2607
Figure 1451. FCCU Event Enable Register (<i>FEE_VD7</i>)	2608
Figure 1452. Event Pending Register (<i>EPR_VD8</i>)	2608
Figure 1453. Reset Event Enable Register (<i>REE_VD8</i>)	2610
Figure 1454. Reset Event Select Register (<i>RES_VD8</i>)	2611
Figure 1455. FCCU Event Enable Register (<i>FEE_VD8</i>)	2612
Figure 1456. Event Pending Register (<i>EPR_VD9</i>)	2613
Figure 1457. Reset Event Enable Register (<i>REE_VD9</i>)	2615
Figure 1458. Reset Event Select Register (<i>RES_VD9</i>)	2616
Figure 1459. FCCU Event Enable Register (<i>FEE_VD9</i>)	2617
Figure 1460. Event Pending Register (<i>EPR_VD10</i>)	2619
Figure 1461. Reset Event Enable Register (<i>REE_VD10</i>)	2620
Figure 1462. FCCU Event Enable Register (<i>FEE_VD10</i>)	2621
Figure 1463. Event Pending Register (<i>EPR_VD12</i>)	2622
Figure 1464. Reset Event Enable Register (<i>REE_VD12</i>)	2623
Figure 1465. Reset Event Select Register (<i>RES_VD12</i>)	2624
Figure 1466. FCCU Event Enable Register (<i>FEE_VD12</i>)	2625
Figure 1467. Event Pending Register (<i>EPR_VD13</i>)	2626
Figure 1468. Reset Event Enable Register (<i>REE_VD13</i>)	2627
Figure 1469. Reset Event Select Register (<i>RES_VD13</i>)	2628
Figure 1470. FCCU Event Enable Register (<i>FEE_VD13</i>)	2629
Figure 1471. Event Pending Register (<i>EPR_VD14</i>)	2630
Figure 1472. Reset Event Enable Register (<i>REE_VD14</i>)	2631
Figure 1473. Reset Event Select Register (<i>RES_VD14</i>)	2632
Figure 1474. FCCU Event Enable Register (<i>FEE_VD14</i>)	2633
Figure 1475. Event Pending Register (<i>EPR_VD15</i>)	2634
Figure 1476. Reset Event Enable Register (<i>REE_VD15</i>)	2636
Figure 1477. Reset Event Select Register (<i>RES_VD15</i>)	2637
Figure 1478. FCCU Event Enable Register (<i>FEE_VD15</i>)	2638
Figure 1479. Voltage Supply for I/O Segments Register (<i>VSIO</i>)	2639
Figure 1480. Event Pending Register (<i>EPR_TD</i>)	2640
Figure 1481. Reset Event Enable Register (<i>REE_TD</i>)	2641
Figure 1482. Reset Event Select Register (<i>RES_TD</i>)	2642
Figure 1483. Temperature Sensor Configuration Register (<i>CTL_TD</i>)	2644
Figure 1484. Temp Sensor FCCU Event Enable Register (<i>FEE_TD</i>)	2645
Figure 1485. Voltage Detect User Mode Test Register (<i>VD_UTST</i>)	2646
Figure 1486. ADC Channel Select Register (<i>ADC_CS</i>)	2646
Figure 1487. Voltage Regulator 1.2V Control Register (<i>VREG1P2_CTRL</i>)	2647

Figure 1488. Module Control Register (MCR)	2648
Figure 1489. MC_PCU block diagram	2653
Figure 1490. Power Domain Status Register (PCU_PSTAT)	2655
Figure 1491. MC_ME block diagram	2657
Figure 1492. Global Status Register (ME_GS)	2672
Figure 1493. Mode Control Register (ME_MCTL)	2674
Figure 1494. Mode Enable Register (ME_ME)	2676
Figure 1495. Interrupt Status Register (ME_IS)	2677
Figure 1496. Interrupt Mask Register (ME_IM)	2678
Figure 1497. Invalid Mode Transition Status Register (ME_IMTS)	2679
Figure 1498. Debug Mode Transition Status Register (ME_DMTS)	2681
Figure 1499. RESET Mode Configuration Register (ME_RESET_MC)	2684
Figure 1500. TEST Mode Configuration Register (ME_TEST_MC)	2685
Figure 1501. SAFE Mode Configuration Register (ME_SAFE_MC)	2685
Figure 1502. DRUN Mode Configuration Register (ME_DRUN_MC)	2686
Figure 1503. RUN0...3 Mode Configuration Registers (ME_RUN0...3_MC)	2686
Figure 1504. HALT0 Mode Configuration Register (ME_HALT0_MC)	2687
Figure 1505. STOP0 Mode Configuration Register (ME_STOP0_MC)	2687
Figure 1506. Peripheral Status Register 0 (ME_PS0)	2689
Figure 1507. Peripheral Status Register 1 (ME_PS1)	2690
Figure 1508. Peripheral Status Register 2 (ME_PS2)	2691
Figure 1509. Peripheral Status Register 3 (ME_PS3)	2692
Figure 1510. Peripheral Status Register 4 (ME_PS4)	2692
Figure 1511. Peripheral Status Register 5 (ME_PS5)	2693
Figure 1512. Peripheral Status Register 6 (ME_PS6)	2693
Figure 1513. Peripheral Status Register 7 (ME_PS7)	2694
Figure 1514. Run Peripheral Configuration Registers (ME_RUN_PC0...7)	2695
Figure 1515. Low-Power Peripheral Configuration Registers (ME_LP_PC0...7)	2696
Figure 1516. Peripheral Control Registers (ME_PCTLn)	2696
Figure 1517. Core Status Register (ME_CS)	2698
Figure 1518. Core Control Register 0 (ME_CCTL0)	2699
Figure 1519. Core Control Register (ME_CCTL1)	2700
Figure 1520. Core Control Register (ME_CCTL2)	2701
Figure 1521. Core Control Register 3 (ME_CCTL3)	2702
Figure 1522. Core Control Register (ME_CCTL4)	2703
Figure 1523. Core Address Register 0 (ME_CADDR0)	2704
Figure 1524. Core Address Register 1 (ME_CADDR1)	2705
Figure 1525. Core Address Register 3 (ME_CADDR3)	2706
Figure 1526. Core Address Register 4 (ME_CADDR4)	2707
Figure 1527. MC_ME Mode Diagram	2708
Figure 1528. MC_ME Transition Diagram	2719
Figure 1529. MC_ME Application Example Flow Diagram	2724
Figure 1530. e200z420n3 debug resources	2728
Figure 1531. DVC1, DVC2 registers	2743
Figure 1532. Debug Control Register 0 (DBCR0) register	2743
Figure 1533. DBCR1 register	2746
Figure 1534. DBCR2 register	2748
Figure 1535. DBCR4 register	2751
Figure 1536. DBCR5 register	2754
Figure 1537. DBCR6 register	2755
Figure 1538. DBCR7 register	2757
Figure 1539. DBCR8 register	2759

Figure 1540. Debug Status Register (DBSR)	2761
Figure 1541. Debug Data Effective Address Register (DDEAR)	2763
Figure 1542. External Debug Resource Allocation Control (EDBRAC0) register	2764
Figure 1543. Debug Event Select register (DEVENT) register	2769
Figure 1544. Debug Data Acquisition Message (DDAM) register	2770
Figure 1545. External Debug Control Register 0 (EDBCR0)	2773
Figure 1546. External Debug Status Register 0 (EDBSR0)	2774
Figure 1547. External Debug Status Register Mask 0 (EDBSRMSK0)	2776
Figure 1548. External Debug Data Effective Address Register (EDDEAR)	2778
Figure 1549. OnCE TAP controller and registers	2779
Figure 1550. OnCE controller implementation	2780
Figure 1551. OnCE Controller and Serial Interface	2784
Figure 1552. OnCE Status register	2785
Figure 1553. OnCE Command (OCMD) register	2786
Figure 1554. OnCE Control Register (OCR)	2790
Figure 1555. CPU Scan Chain Register (CPUSCR)	2795
Figure 1556. Control State Register (CTL)	2796
Figure 1557. e200z425n3 debug resources	2809
Figure 1558. DVC1, DVC2 registers	2824
Figure 1559. Debug Control Register 0 (DBCR0) register	2824
Figure 1560. DBCR1 register	2827
Figure 1561. DBCR2 register	2829
Figure 1562. DBCR4 register	2832
Figure 1563. DBCR5 register	2835
Figure 1564. DBCR6 register	2836
Figure 1565. DBCR7 register	2838
Figure 1566. DBCR8 register	2840
Figure 1567. Debug Status Register (DBSR)	2842
Figure 1568. Debug Data Effective Address Register (DDEAR)	2844
Figure 1569. External Debug Resource Allocation Control (EDBRAC0) register	2845
Figure 1570. Debug Event Select register (DEVENT) register	2851
Figure 1571. Debug Data Acquisition Message (DDAM) register	2852
Figure 1572. External Debug Control Register 0 (EDBCR0)	2855
Figure 1573. External Debug Status Register 0 (EDBSR0)	2856
Figure 1574. External Debug Status Register Mask 0 (EDBSRMSK0)	2858
Figure 1575. External Debug Data Effective Address Register (EDDEAR)	2860
Figure 1576. OnCE TAP controller and registers	2861
Figure 1577. OnCE controller implementation	2862
Figure 1578. OnCE Controller and Serial Interface	2866
Figure 1579. OnCE Status register	2867
Figure 1580. OnCE Command (OCMD) register	2868
Figure 1581. OnCE Control Register (OCR)	2872
Figure 1582. CPU Scan Chain Register (CPUSCR)	2877
Figure 1583. Control State Register (CTL)	2878
Figure 1584. DCI block diagram	2889
Figure 1585. DCI in a standard configuration	2890
Figure 1586. DCI operating modes state diagram	2891
Figure 1587. DCI JTAG/LFAST switch integration	2893
Figure 1588. DCI switching between JTAG and LFAST modes	2894
Figure 1589. DCI JTAG to LFAST switching diagram	2895
Figure 1590. DCI LFAST to JTAG switching diagram	2896
Figure 1591. DCI debug control mechanism	2899

Figure 1592.DCI debug signal generation	2900
Figure 1593.DCI EVTO management.	2901
Figure 1594.DCI EVTO timing diagram	2902
Figure 1595.DCI control register (DCI_CR)	2903
Figure 1596.JTAG (IEEE 1149.1) block diagram	2908
Figure 1597.6-bit instruction register	2911
Figure 1598.Device identification register.	2912
Figure 1599.JTAG_PASSWORD register.	2912
Figure 1600.Shifting data through a register.	2913
Figure 1601.IEEE 1149.1-2001 TAP controller finite state machine.	2914
Figure 1602.TAP.7 Controller Architecture	2924
Figure 1603.TAP.7 Behaviors.	2924
Figure 1604.CJTAG operating model	2926
Figure 1605.Conceptual view of the EPU reset logic and state machines	2928
Figure 1606.Reset state machine.	2929
Figure 1607.Conceptual view of technology selection mechanism	2931
Figure 1608.Falling edge TAP controller state machine.	2933
Figure 1609.DR_Scan sequence used for command creation	2946
Figure 1610.EPU operating states	2952
Figure 1611.Conceptual block diagram of the APU functions	2955
Figure 1612.Conceptual view of APU operation	2956
Figure 1613.Scan Packet sequence.	2958
Figure 1614.Serialization and De-serialization of Scan Packet Information	2959
Figure 1615.CPA State Entry from BPA State with Continued Online Operation	2960
Figure 1616.CPA State Entry from the SPA State with Continued Online Operation	2961
Figure 1617.CPA State Entry from the BPA State with Offline Operation	2962
Figure 1618.CPA State Entry from the SPA State with Offline Operation	2963
Figure 1619.MScan Transaction.	2967
Figure 1620.OScan Scan Packet content/TAPC state relationships	2968
Figure 1621.JDC block diagram	2971
Figure 1622.Module Configuration Register (MCR)	2971
Figure 1623.Module Status Register (MSR)	2972
Figure 1624.JOUT_IPS register	2972
Figure 1625.JIN_IPS register	2973
Figure 1626.JOUT JTAG register	2973
Figure 1627.JIN JTAG Register	2974
Figure 1628.SPU block diagram	2977
Figure 1629.SPU module interface.	2982
Figure 1630.L1SEL0 register format.	2987
Figure 1631.L1SEL1 register format.	2988
Figure 1632.L1SEL2 register format.	2988
Figure 1633.L1SEL3 register format.	2988
Figure 1634.L1SEL4 register format.	2989
Figure 1635.L1SEL5 register format.	2989
Figure 1636.L1SEL6 register format.	2989
Figure 1637.L1SEL7 register format.	2990
Figure 1638.ITLD2 register format	2990
Figure 1639.ITLD3 register format	2991
Figure 1640.COPEVP register format	2991
Figure 1641.C1PEVP register format	2992
Figure 1642.C2PEVP register format	2992
Figure 1643.COPIS register format	2993

Figure 1644.C1PIS register format	2993
Figure 1645.C2PIS register format	2994
Figure 1646.Level2 Mux AND gates input pin routing for State0	2995
Figure 1647.L2nSEL0 register format	2996
Figure 1648.L2nSEL1 register format	2997
Figure 1649.L2nSEL2 register format	2998
Figure 1650.L2nSEL3 register format	3000
Figure 1651.IOICn register format	3001
Figure 1652.SCTRL register format	3003
Figure 1653.SnGC register format	3005
Figure 1654.SS register format	3007
Figure 1655.SE register format	3009
Figure 1656.SnTA0 register format	3013
Figure 1657.SnTA1 register format	3014
Figure 1658.SnFA0 register format	3015
Figure 1659.SnFA1 register format	3016
Figure 1660.Trace group configuration register format	3017
Figure 1661.INTS register format	3018
Figure 1662.CCTRLn register format	3018
Figure 1663.CCMPn register format	3019
Figure 1664.CCOMS register format	3020
Figure 1665.COS register format	3021
Figure 1666.CCAPS register format	3021
Figure 1667.Input Mux unit block diagram	3023
Figure 1668.SLU block diagram	3025
Figure 1669.Example of a simple configurable if-then-else sequence	3026
Figure 1670.Complex sequence example	3027
Figure 1671.Counters/timers overview	3028
Figure 1672.Input selection for count increment	3029
Figure 1673.SPU counters increment assignment	3029
Figure 1674.CPU PMC mapping inside the SPU unit	3030
Figure 1675.JTAGM block diagram	3032
Figure 1676.LFAST-to-DCI flow diagram	3033
Figure 1677.JTAGM to DCI serial interface diagram	3036
Figure 1678.Bit ordering for LFAST to JTAGM transfer	3037
Figure 1679.Bit ordering for JTAGM to LFAST transfer	3037
Figure 1680.JTAGM_MCR register	3038
Figure 1681.JTAGM_SR register in LFAST mode	3040
Figure 1682.JTAGM_SR register in SW mode	3041
Figure 1683.JTAGM_DOR0 register	3043
Figure 1684.JTAGM_DOR1 register	3043
Figure 1685.JTAGM_DOR2 register	3044
Figure 1686.JTAGM_DOR3 register	3044
Figure 1687.JTAGM_RxCRC register	3045
Figure 1688.JTAGM_DIR0 register	3045
Figure 1689.JJTAGM_DIR1 register	3046
Figure 1690.LFAST block diagram	3047
Figure 1691.LFAST frame structure	3049
Figure 1692.Serial frame structure	3050
Figure 1693.Key to Register Fields	3053
Figure 1694.Mode Configuration Register (MCR)	3054
Figure 1695.Speed Control Register (SCR)	3056

Figure 1696. Correlator Control Register (COCR)	3057
Figure 1697. Test Mode Control Register (TMCR)	3058
Figure 1698. Auto Loopback Control Register (ALCR)	3059
Figure 1699. Rate Change Delay Control Register (RCDCR)	3060
Figure 1700. Wakeup Delay Control Register (SLCR)	3061
Figure 1701. Ping Control Register (PICR)	3063
Figure 1702. LVDS Control Register (LCR)	3064
Figure 1703. Unsolicited Tx Control Register (UNSTCR)	3066
Figure 1704. Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)	3067
Figure 1705. Global Status Register (GSR)	3067
Figure 1706. Data frame Status Register (DFSR)	3069
Figure 1707. Tx Interrupt Status Register (TISR)	3070
Figure 1708. Rx Interrupt Status Register (RISR)	3071
Figure 1709. Rx ICLC Interrupt Status Register (RIISR)	3072
Figure 1710. PLL and LVDS Status Register (PLLLSR)	3074
Figure 1711. Unsolicited Rx Status Register (UNSRSR)	3075
Figure 1712. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)	3075
Figure 1713. Top Level Receive Controller	3078
Figure 1714. Hunt Correlation mode	3079
Figure 1715. High Speed 8 Phase Selection	3079
Figure 1716. High Speed 8 Phase Clock Alignment Example	3080
Figure 1717. High Speed 4 Phase Clock Alignment Example	3080
Figure 1718. Low speed 4 phase clock alignment example	3081
Figure 1719. Samplers, each Sampler has 3 Registers	3083
Figure 1720. Sampler 5 Logic	3084
Figure 1721. Extracting Header and Payload from the 16-bit output from Auto correlation.	3085
Figure 1722. Transmit Controller Connections	3087
Figure 1723. Example of Sleep mode	3089
Figure 1724. Rx LoopBack mode	3094
Figure 1725. Rx LVDS LoopBack mode	3095
Figure 1726. Tx LoopBack mode without automatic frame generation	3096
Figure 1727. Automatic Loopback Test status signal timings	3097
Figure 1728. Tx LoopBack mode with automatic frame generation.	3098
Figure 1729. Tx LVDS Loopback (external) mode with automatic frame generation.	3099
Figure 1730. LFAST module clock domains	3102
Figure 1731. External clock muxing of lfast_sysclk	3103
Figure 1732. External clock muxing of lfast_sysclk of 20/26 MHz	3104
Figure 1733. External clock muxing of lfast_sysclk of 10 MHz	3104
Figure 1734. Slow speed 4 phases generation	3105
Figure 1735. Clock Enables and Clock Paths	3105
Figure 1736. Transmit Clocks Muxing	3107
Figure 1737. DTS block diagram	3109
Figure 1738. DTS event sequence example	3110
Figure 1739. DTS device connections	3110
Figure 1740. DTS_ENABLE register	3112
Figure 1741. DTS_STARTUP register	3113
Figure 1742. DTS_SEMAPHORE register	3113
Figure 1743. DTS_SEMAPHORE_B register	3114
Figure 1744. DTS startup sequence example	3116
Figure 1745. NAR top level block diagram	3117
Figure 1746. NAR control register (NAR_CR)	3119
Figure 1747. NAR status register (NAR_ST)	3121

Figure 1748. Source filter register (NAR_SFR)	3123
Figure 1749. Message type filter register (NAR_TFR).	3124
Figure 1750. Trace memory base address register NAR_TBAHI	3126
Figure 1751. Trace memory base address register NAR_TBALO	3126
Figure 1752. Trace memory bus control register (NAR_TCR).	3127
Figure 1753. Suppress trigger control registers (NAR_STCR1–4)	3128
Figure 1754. Client suppress status register (NAR_CSSR).	3129
Figure 1755. Client disable register (NAR_CDR).	3130
Figure 1756. Trace memory fill level and partition configuration register (NAR_AHFPAR).	3132
Figure 1757. AUX bus parsing.	3136
Figure 1758. Timestamp architecture	3145
Figure 1759. Nexus controller state machine	3148
Figure 1760. Shifting data into register	3148
Figure 1761. Nexus command format for register selection	3149
Figure 1762. Legacy client Interface controller	3153
Figure 1763. Nexus 3 functional block diagram	3157
Figure 1764. Client Select Control (CSC) register	3165
Figure 1765. Port Configuration Register (PCR)	3166
Figure 1766. Development Control 1 (DC1) register	3167
Figure 1767. Development Control 2 (DC2) register	3168
Figure 1768. Development Control 3 (DC3) register	3170
Figure 1769. Development Control 4 (DC4) register	3173
Figure 1770. Development Status (DS) register	3174
Figure 1771. Watchpoint Trigger (WT) register	3175
Figure 1772. Program Trace Start Trigger Control (PTSTC) register	3176
Figure 1773. Program Trace End Trigger Control (PTETC) register	3177
Figure 1774. Data Trace Start Trigger Control (DTSTC) register	3178
Figure 1775. Data Trace End Trigger Control (DTETC) register	3179
Figure 1776. Watchpoint Mask (WMSK) register.	3180
Figure 1777. Nexus Overrun Control (OVCR) register	3182
Figure 1778. OVCR field descriptions	3182
Figure 1779. Data Trace Control (DTC) register	3183
Figure 1780. Data Trace Start Address 1 (DTSA1) register	3184
Figure 1781. Data Trace Start Address 2 (DTSA2) register	3184
Figure 1782. Data Trace Start Address 3 (DTSA3) register	3185
Figure 1783. Data Trace Start Address 4 (DTSA4) register	3185
Figure 1784. Data Trace End Address 1 register	3185
Figure 1785. Data Trace End Address 2 register	3185
Figure 1786. Data Trace End Address 3 register	3185
Figure 1787. Data Trace End Address 4 register	3185
Figure 1788. Read/Write Access Control/Status (RWCS) register	3186
Figure 1789. Read/Write Access Data (RWD) register	3188
Figure 1790. Read/Write Access Address (RWA) register	3189
Figure 1791. JTAG DR format	3190
Figure 1792. Relative address generation and recreation	3192
Figure 1793. Debug Status message format	3195
Figure 1794. Error message format	3196
Figure 1795. Nexus 3 Process ID Register (NPIDR)	3197
Figure 1796. Ownership Trace Message format	3197
Figure 1797. Indirect branch message (history) format	3199
Figure 1798. Indirect branch message format	3200
Figure 1799. Direct branch message format	3200

Figure 1800.Resource Full Message format	3202
Figure 1801.Program Correlation message formats	3203
Figure 1802.Program Trace Synchronization message format	3204
Figure 1803.Direct/Indirect Branch with Sync message format	3205
Figure 1804.Indirect Branch History w/ Sync message format	3205
Figure 1805.Program Trace — Indirect Branch message (Traditional)	3206
Figure 1806.Program Trace — Indirect Branch message (history)	3207
Figure 1807.Program Trace — Direct Branch (traditional) & Error messages	3207
Figure 1808.Program Trace — Indirect Branch w/ Sync message	3207
Figure 1809.Data Write message format	3208
Figure 1810.Data Read message format	3208
Figure 1811.Data Write/Read with Sync message format	3209
Figure 1812.Data Trace — Data Write message	3212
Figure 1813.Data Trace — Data Read w/ Sync message	3212
Figure 1814.Data Acquisition message format	3213
Figure 1815.Watchpoint message format	3214
Figure 1816.Watchpoint Message & Watchpoint Error message	3215
Figure 1817.Error message format	3220
Figure 1818.Single pin MSEO transfers	3224
Figure 1819.Dual pin MSEO transfers	3225
Figure 1820.Nexus 3 functional block diagram	3234
Figure 1821.Client Select Control (CSC) register	3243
Figure 1822.Port Configuration Register (PCR)	3243
Figure 1823.Development Control 1 (DC1) register	3244
Figure 1824.Development Control 2 (DC2) register	3246
Figure 1825.Development Control 3 (DC3) register	3248
Figure 1826.Development Control 4 (DC4) register	3251
Figure 1827.Development Status (DS) register	3252
Figure 1828.Watchpoint Trigger (WT) register	3253
Figure 1829.Program Trace Start Trigger Control (PTSTC) register	3254
Figure 1830.Program Trace End Trigger Control (PTETC) register	3255
Figure 1831.Data Trace Start Trigger Control (DTSTC) register	3256
Figure 1832.Data Trace End Trigger Control (DTETC) register	3257
Figure 1833.Watchpoint Mask (WMSK) register	3258
Figure 1834.Nexus Overrun Control (OVCR) register	3260
Figure 1835.OVCR field descriptions	3260
Figure 1836.Data Trace Control (DTC) register	3261
Figure 1837.Data Trace Start Address 1 (DTSA1) register	3262
Figure 1838.Data Trace Start Address 2 (DTSA2) register	3262
Figure 1839.Data Trace Start Address 3 (DTSA3) register	3263
Figure 1840.Data Trace Start Address 4 (DTSA4) register	3263
Figure 1841.Data Trace End Address 1 register	3263
Figure 1842.Data Trace End Address 2 register	3263
Figure 1843.Data Trace End Address 3 register	3263
Figure 1844.Data Trace End Address 4 register	3263
Figure 1845.Read/Write Access Control/Status (RWCS) register	3264
Figure 1846.Read/Write Access Data (RWD) register	3266
Figure 1847.Read/Write Access Address (RWA) register	3267
Figure 1848.JTAG DR format	3268
Figure 1849.Relative address generation and recreation	3270
Figure 1850.Debug Status message format	3273
Figure 1851.Error message format	3274

Figure 1852. Nexus 3 Process ID Register (NPIDR)	3275
Figure 1853. Ownership Trace Message format	3275
Figure 1854. Indirect branch message (history) format	3278
Figure 1855. Indirect branch message format	3278
Figure 1856. Direct branch message format	3278
Figure 1857. Resource Full Message format	3280
Figure 1858. Program Correlation message formats	3281
Figure 1859. Program Trace Synchronization message format	3282
Figure 1860. Direct/Indirect Branch with Sync message format	3284
Figure 1861. Indirect Branch History w/ Sync message format	3284
Figure 1862. Program Trace — Indirect Branch message (Traditional)	3285
Figure 1863. Program Trace — Indirect Branch message (history)	3285
Figure 1864. Program Trace — Direct Branch (traditional) & Error messages	3286
Figure 1865. Program Trace — Indirect Branch w/ Sync message	3286
Figure 1866. Data Write message format	3287
Figure 1867. Data Read message format	3287
Figure 1868. Data Write/Read with Sync message format	3288
Figure 1869. Data Trace — Data Write message	3291
Figure 1870. Data Trace — Data Read w/ Sync message	3291
Figure 1871. Data Acquisition message format	3292
Figure 1872. Watchpoint message format	3293
Figure 1873. Watchpoint Message & Watchpoint Error message	3294
Figure 1874. Error message format	3299
Figure 1875. Single pin MSEO transfers	3302
Figure 1876. Dual pin MSEO transfers	3303
Figure 1877. GTMDI block diagram	3309
Figure 1878. GTMDI and GTM interface	3310
Figure 1879. Watchpoint triggers and watchpoint messages	3312
Figure 1880. Device identity register (DID)	3318
Figure 1881. GTMDI development control register (GTMDI_DC)	3319
Figure 1882. GTMDI development status register (GTMDI_DS)	3322
Figure 1883. TIM watchpoint control 1 register (GTMDI_TIM_WPC1)	3324
Figure 1884. TIM watchpoint control 2 register (GTMDI_TIM_WPC2)	3326
Figure 1885. TOM watchpoint control 1 register (GTMDI_TOM_WPC1)	3328
Figure 1886. TOM watchpoint control 2 register (GTMDI_TOM_WPC2)	3330
Figure 1887. ATOM watchpoint control 1 register (GTMDI_ATOM_WPC1)	3332
Figure 1888. ATOM watchpoint control 2 register (GTMDI_ATOM_WPC2)	3335
Figure 1889. SPEA watchpoint control 1 register (GTMDI_SPEA_WPC1)	3337
Figure 1890. SPEA watchpoint control 2 register (GTMDI_SPEA_WPC2)	3338
Figure 1891. SPEB watchpoint control 1 register (GTMDI_SPEB_WPC1)	3340
Figure 1892. SPEB watchpoint control 2 register (GTMDI_SPEB_WPC2)	3342
Figure 1893. DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1)	3344
Figure 1894. DPLL watchpoint control 2 register (GTMDI_DPLL_WPC2)	3345
Figure 1895. DPLL watchpoint control 3 register (GTMDI_DPLL_WPC3)	3347
Figure 1896. DPLL watchpoint control 4 register (GTMDI_DPLL_WPC4)	3348
Figure 1897. DPLL watchpoint control 5 register (GTMDI_DPLL_WPC5)	3348
Figure 1898. DPLL data trace control register (GTMDI_DPLL_DTC)	3349
Figure 1899. ARU watchpoint control 1 register (GTMDI_ARU_WPC1)	3350
Figure 1900. ARU watchpoint control 2 register (GTMDI_ARU_WPC2)	3352
Figure 1901. ARU watchpoint DATA0H register (GTMDI_ARU_DATA0H)	3353
Figure 1902. ARU watchpoint DATA0L register (GTMDI_ARU_DATA0L)	3354
Figure 1903. ARU watchpoint DATA1H register (GTMDI_ARU_DATA1H)	3354

Figure 1904. ARU Watchpoint DATA1L (GTMDI_ARU_DATA1L)	3355
Figure 1905. ARU data trace control register (GTMDI_ARU_DTC)	3356
Figure 1906. MCSA/B development control register (GTMDI_MCSA_DC and GTMDI_MCSB_DC)	3357
Figure 1907. MCSA/B watchpoint control register (GTMDI_MCSA_WPC and GTMDI_MCSB_WPC)	3358
Figure 1908. MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC)	3360
Figure 1909. MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC)	3361
Figure 1910. MCSA/B watchpoint address 1 register (GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1)	3363
Figure 1911. MCSA/B watchpoint address 2 register (GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2)	3364
Figure 1912. MCSA/B watchpoint data 1 register (GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1)	3364
Figure 1913. MCSA/B watchpoint data 2 register (GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2)	3365
Figure 1914. MCSA/B channel enable register (GTMDI_MCSA_CE and GTMDI_MCSB_CE)	3366
Figure 1915. Data trace address range 1 register (GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1)	3368
Figure 1916. MCSA data trace address range 2 register (GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2)	3368
Figure 1917. TBU0 watchpoint control 1 register (GTMDI_TBU0_WPC1)	3369
Figure 1918. TBU0 watchpoint control 2 register (GTMDI_TBU0_WPC2)	3371
Figure 1919. TBU0 watchpoint DATA register (GTMDI_TBU0_DATA)	3372
Figure 1920. TBU1 watchpoint control 1 register (GTMDI_TBU1_WPC1)	3373
Figure 1921. TBU1 watchpoint control 2 register (GTMDI_TBU1_WPC2)	3374
Figure 1922. TBU1 watchpoint DATA register (GTMDI_TBU1_DATA)	3376
Figure 1923. TBU2 watchpoint control 1 register (GTMDI_TBU2_WPC1)	3377
Figure 1924. TBU2 watchpoint control 2 register (GTMDI_TBU2_WPC2)	3378
Figure 1925. TBU2 watchpoint DATA register (GTMDI_TBU2_DATA)	3380
Figure 1926. GTM Halt signal enable logic	3381
Figure 1927. TIM selection logic	3382
Figure 1928. Shifting data into register	3388
Figure 1929. IEEE 1149.1 16-State finite state machine	3389
Figure 1930. Nexus controller state machine	3390
Figure 1931. IEEE 1149.1 controller command input	3391
Figure 1932. Debug status message format	3392
Figure 1933. Debug status error message format	3392
Figure 1934. Data write message format	3393
Figure 1935. Data read message format	3393
Figure 1936. GTMDI generic data trace flow diagram	3394
Figure 1937. Data trace address range options	3395
Figure 1938. Fetch trace state machine state diagram	3396
Figure 1939. Watchpoint hit message format	3397
Figure 1940. Development control register 1 (DC1)	3404
Figure 1941. Development control register 2 (DC2)	3405
Figure 1942. Watchpoint trigger register (WT)	3406
Figure 1943. Data trace control (DTC) register	3407
Figure 1944. Data trace start address registers 1 and 2 (DTSA1 and DTSA2)	3408
Figure 1945. Data trace end address registers 1 and 2 (DTEA1 and DTEA2)	3408
Figure 1946. Breakpoint/watchpoint control register 1	3409
Figure 1947. Breakpoint/watchpoint control register 2	3410
Figure 1948. Breakpoint /watchpoint address registers 1 and 2	3411

Figure 1949. JTAG DR for Nexus register access	3412
Figure 1950. DTM message format for DATA width = 64 bits	3413
Figure 1951. DTM error message format (fixed length = 48 bits)	3414
Figure 1952. Data write/read with synchronization message format	3414
Figure 1953. Relative addressing example	3416
Figure 1954. Watchpoint message format (fixed length = 18 bits).	3418
Figure 1955. Error message format (fixed length = 28 bits).	3418
Figure 1956. Peripheral DATA bus fields.	3419
Figure 1957. Peripheral message format (max length = 60 bits, min length = 17 bits)	3419
Figure 1958. Error message format (fixed length = 28 bits).	3419
Figure 1959. Message with TSTAMP field example	3420
Figure 1960. CRC top level diagram	3421
Figure 1961. CRC Configuration Register (CRC_CFG).	3423
Figure 1962. CRC Input Register (CRC_INP)	3424
Figure 1963. CRC Current Status Register (CRC_CSTAT).	3425
Figure 1964. CRC Output Register (CRC_OUTP).	3426
Figure 1965. CRC-CCITT engine concept scheme	3427
Figure 1966. CRC computation flow	3428
Figure 1967. DMA-CRC transmission sequence	3430
Figure 1968. MEMU block diagram	3432
Figure 1969. MEMU error buffer uniqueness check flow chart	3435
Figure 1970. MEMU reporting table uniqueness check flow chart.	3436
Figure 1971. Control register (CTRL)	3439
Figure 1972. Error flag register (ERR_FLAG)	3440
Figure 1973. Debug register (DEBUG)	3442
Figure 1974. System RAM correctable error reporting table status register (SYS_RAM_CERR_STS)	3444
Figure 1975. System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDRn)	3445
Figure 1976. System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)	3445
Figure 1977. System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)	3446
Figure 1978. System RAM concurrent overflow registers (SYS_RAM_OFLWn).	3447
Figure 1979. Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS)	3447
Figure 1980. Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDRn)	3448
Figure 1981. Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)	3449
Figure 1982. Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)	3450
Figure 1983. Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLWn)	3450
Figure 1984. Flash memory correctable error reporting table status register (FLASH_CERR_STS)	3451
Figure 1985. Flash memory correctable error reporting table address register (FLASH_CERR_ADDRn)	3452
Figure 1986. Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)	3452
Figure 1987. Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)	3453
Figure 1988. Flash memory concurrent overflow registers (FLASH_OFLWn).	3454
Figure 1989. Generic representation of the reporting table of one MEMU reporting block	3456

Figure 1990. Cache Debug Access Control Register (CDACNTL)	3459
Figure 1991. Cache Debug Access Data Register (CDADATA)	3460
Figure 1992. e2eECC Error Control/Status Register 0 (E2EECSR0)	3461
Figure 1993. IMA block diagram	3463
Figure 1994. IMA Control Register (IMA_CTRL)	3465
Figure 1995. IMA Enable Access Register (IMA_ENABLE)	3466
Figure 1996. IMA Status Register (IMA_STATUS)	3467
Figure 1997. IMA RAM SELECT Register (IMA_SLCT)	3468
Figure 1998. IMA Write Unlock Register (IMA_WRITE_UNLOCK)	3469
Figure 1999. IMA Read Unlock Register (IMA_READ_UNLOCK)	3469
Figure 2000. IMA RAM Write Data Register n (IMA_WRITE_DATA_n)	3470
Figure 2001. IMA READ_DATA_n Register (IMA_READ_DATA_n)	3471
Figure 2002. FCCU block diagram	3477
Figure 2003. FCCU state diagram	3480
Figure 2004. Recoverable FAULT (ALARM state) recovery (a)	3482
Figure 2005. Recoverable FAULT (ALARM state) recovery (b)	3483
Figure 2006. Recoverable FAULT (ALARM -> FAULT state) recovery	3484
Figure 2007. FAULT injection	3485
Figure 2008. FAULT clearing	3485
Figure 2009. STCU-FCCU (case a)	3486
Figure 2010. STCU-FCCU (case b)	3486
Figure 2011. STCU-FCCU (case c)	3486
Figure 2012. Dual-rail protocol (slow switching mode)	3489
Figure 2013. Dual-rail protocol (fast switching mode)	3489
Figure 2014. Time-switching protocol	3490
Figure 2015. Bi-stable protocol	3491
Figure 2016. Error signal flow	3492
Figure 2017. FCCU Control Register (FCCU_CTRL)	3496
Figure 2018. FCCU CTRL Key Register (FCCU_CTRLK)	3498
Figure 2019. FCCU Configuration Register (FCCU_CFG)	3499
Figure 2020. FCCU RF Configuration Register [0:3] (FCCU_RF_CFG[0:3])	3501
Figure 2021. FCCU RFS Configuration Register (FCCU_RFS_CFG[0:7])	3502
Figure 2022. FCCU RF Status Register (FCCU_RF_S[0:3])	3504
Figure 2023. FCCU RF Key Register (FCCU_RFK)	3505
Figure 2024. FCCU RF Enable Register (FCCU_RF_E[0:3])	3506
Figure 2025. FCCU RF Timeout Enable Register (FCCU_RF_TOE[0:3])	3507
Figure 2026. FCCU RF Time-out Register (FCCU_RF_TO)	3508
Figure 2027. FCCU CFG Timeout Register (FCCU_CFG_TO)	3509
Figure 2028. FCCU IO Control Register (FCCU_EINOUT)	3510
Figure 2029. FCCU Status Register (FCCU_STAT)	3511
Figure 2030. FCCU NA Freeze Status Register (N2AF_STATUS)	3512
Figure 2031. FCCU AF Freeze Status Register (A2FF_STATUS)	3513
Figure 2032. FCCU NF Freeze Status Register (N2FF_STATUS)	3515
Figure 2033. FCCU FA Freeze Status Register (F2A_STATUS)	3516
Figure 2034. FCCU RF Fake Register (FCCU_RFF)	3517
Figure 2035. FCCU IRQ Status Register (FCCU_IRQ_STAT)	3518
Figure 2036. FCCU IRQ Enable Register (FCCU_IRQ_EN)	3519
Figure 2037. FCCU XTMR Register (FCCU_XTMR)	3520
Figure 2038. FCCU MCS Register (FCCU_MCS)	3520
Figure 2039. FCCU Transient Register (FCCU_TRANS_LOCK)	3521
Figure 2040. FCCU Transient Register (FCCU_PERMNT_LOCK)	3522
Figure 2041. FCCU Delta T Register (FCCU_DELTA_T)	3522

Figure 2042.FCCU IRQ Alarm Enable Registers (FCCU_IRQ_ALARM_EN[0:3])	3523
Figure 2043.FCCU NMI Enable Registers (FCCU_NMI_EN[0:3])	3524
Figure 2044.FCCU EOUT Signaling Enable Registers (FCCU_EOUT_SIG_EN[0:3])	3525
Figure 2045.FOSU connections to the FCCU and MC_RGM	3530
Figure 2046.STCU2 block diagram	3534
Figure 2047.Register figure conventions	3541
Figure 2048.STCU2 Run Register (STCU_RUN)	3542
Figure 2049.STCU2 Run Software Register (STCU_RUNSW)	3543
Figure 2050.STCU2 SK Code Register (STCU_SKC)	3545
Figure 2051.STCU2 Configuration Register (STCU_CFG)	3546
Figure 2052.STCU2 PLL Configuration Register (STCU_PLL_CFG)	3548
Figure 2053.STCU2 Watchdog Register Granularity (STCU_WDG)	3549
Figure 2054.STCU2 Interrupt Flag Register (STCU_INT_FLG)	3550
Figure 2055.STCU2 CRC Expected Status Register (STCU_CRCE)	3551
Figure 2056.STCU2 CRC Read Status Register (STCU_CRCR)	3552
Figure 2057.STCU2 Error Register (STCU_ERR_STAT)	3553
Figure 2058.STCU2 Error FM Register (STCU_ERR_FM)	3556
Figure 2059.STCU2 Off-Line LBIST Status Register (STCU_LBS)	3557
Figure 2060.STCU2 Off-Line LBIST End Flag Register (STCU_LBE)	3558
Figure 2061.STCU2 On-Line LBIST Status Register (STCU_LBSSW)	3558
Figure 2062.STCU2 On-Line LBIST End Flag Register (STCU_LBESW)	3559
Figure 2063.STCU2 On-Line LBIST Reset Management (STCU_LBRMSW)	3560
Figure 2064.STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)	3561
Figure 2065.STCU2 Off-Line MBIST Status Low Register (STCU_MBSL)	3561
Figure 2066.STCU2 Off-Line MBIST Status Medium Register (STCU_MBSM)	3562
Figure 2067.STCU2 Off-Line MBIST Status High Register (STCU_MBSH)	3563
Figure 2068.STCU2 Off-Line MBIST End Flag Low Register (STCU_MBEL)	3564
Figure 2069.STCU2 Off-Line MBIST End Flag Medium Register (STCU_MBEM)	3564
Figure 2070.STCU2 Off-Line MBIST End Flag High Register (STCU_MBEH)	3565
Figure 2071.STCU2 On-Line MBIST Status Low Register (STCU_MBSLSW)	3566
Figure 2072.STCU2 On-Line MBIST Status Medium Register (STCU_MBSMSW)	3567
Figure 2073.STCU2 On-Line MBIST Status High Register (STCU_MBSHSW)	3568
Figure 2074.STCU2 On-Line MBIST End Flag Low Register (STCU_MBELSW)	3569
Figure 2075.STCU2 On-Line MBIST End Flag Medium Register (STCU_MBEMSW)	3570
Figure 2076.STCU2 On-Line MBIST End Flag High Register (STCU_MBEHSW)	3571
Figure 2077.STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)	3572
Figure 2078.STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)	3573
Figure 2079.STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)	3574
Figure 2080.STCU2 LBIST Control Register (STCU_LB_CTRL)	3575
Figure 2081.STCU2 LBIST PC Stop Register (STCU_LB_PCS)	3577
Figure 2082.STCU2 LBIST PRPG Low Register (STCU_LB_PRPGL)	3578
Figure 2083.STCU2 LBIST PRPG High Register (STCU_LB_PRPGH)	3579
Figure 2084.STCU2 Off-Line LBIST MISR Expected Low Register (STCU_LB_MISREL)	3580
Figure 2085.STCU2 LBIST MISR Expected High Register (STCU_LB_MISREH)	3580
Figure 2086.STCU2 Off-Line LBIST MISR Read Low Register (STCU_LB_MISRRL)	3581
Figure 2087.STCU2 Off-Line LBIST MISR Read High Register (STCU_LB_MISRRH)	3582
Figure 2088.STCU2 On-Line LBIST MISR Expected Low Register (STCU_LB_MISRELSW)	3583
Figure 2089.STCU2 On-Line LBIST MISR Expected High Register (STCU_LB_MISREHSW)	3584
Figure 2090.STCU2 On-Line LBIST MISR Read Low Register (STCU_LB_MISRRLSW)	3585
Figure 2091.STCU2 On-Line LBIST MISR Read High Register (STCU_LB_MISRRLHSW)	3586
Figure 2092.STCU2 MBIST Control Register (STCU_MB_CTRL)	3587
Figure 2093.REG_PROT Block Diagram	3598

Figure 2094. REG_PROT Memory Diagram	3599
Figure 2095. Key to Register Fields.	3601
Figure 2096. Soft Lock Bit Register (SLBRn)	3601
Figure 2097. Global Configuration Register (GCR)	3603
Figure 2098. Change lock settings directly via Area #4.	3604
Figure 2099. Change lock settings for 16-bit protected addresses	3605
Figure 2100. Change lock settings for 32-bit protected addresses	3605
Figure 2101. Change lock settings for mixed protection	3605
Figure 2102. Enable Locking Via Mirror Module Space (Area #3).	3606
Figure 2103. Enable locking for protected and unprotected addresses.	3606
Figure 2104. PASS module block diagram	3609
Figure 2105. Life Cycle Status register (LCSTAT)	3612
Figure 2106. CHSEL register	3613
Figure 2107. CHSTAT register	3614
Figure 2108. CINn register	3615
Figure 2109. Production Disable DCF client definition	3617
Figure 2110. Clock Jitter Enable	3618
Figure 2111. LOCK0_PGn register	3619
Figure 2112. LOCK1_PGn register	3620
Figure 2113. LOCK2_PGn register	3622
Figure 2114. LOCK3_PGn Register	3623
Figure 2115. Tamper Detection Module block diagram	3631
Figure 2116. Key for register field	3633
Figure 2117. Diary Base Address (DBA) DCF client	3634
Figure 2118. Tamper Region Override (TO) DCF client	3635
Figure 2119. One Time Programmable Enable (OTPEN0) DCF client	3635
Figure 2120. One Time Programmable Enable (OTPEN1) DCF client	3636
Figure 2121. One Time Programmable Enable (OTPEN2) DCF client	3637
Figure 2122. One Time Programmable Enable (OTPEN3) DCF client	3638
Figure 2123. Tamper Region Assignment DCF client (TDRn_LOCK0).	3639
Figure 2124. Tamper Region Assignment DCF client (TDRn_LOCK1)	3640
Figure 2125. Tamper Region Assignment DCF client (TDRn_LOCK2)	3641
Figure 2126. Tamper Region Assignment DCF client (TDRn_LOCK3)	3641
Figure 2127. TDR Status and Diary Override Register (TDRSR)	3642
Figure 2128. Last Flash Programmed Address Register (LFPAR)	3643
Figure 2129. Diary implementation	3645
Figure 2131. Key to register fields	3651
Figure 2132. AMU V Pointer Channel n register (VCHn).	3651
Figure 2133. AMU U Pointer Channel n register (UCHn)	3652
Figure 2134. P1 Pointer Channel n register (P1CHn)	3653
Figure 2135. AMU P2 Pointer Channel n register (P2CHn).	3653
Figure 2136. AMU P3 Pointer Channel n register (P3CHn).	3654
Figure 2137. P8 Channel n register (P8CHn).	3655
Figure 2138. AMU P6 Channel n register (P6CHn)	3655
Figure 2139. AMU Ncrossing Channel n register (NCHn)	3656
Figure 2140. AMU P7 Channel n register (P7CHn)	3657
Figure 2141. AMU P4, P5 Channel n register	3657
Figure 2142. AMU Result Channel n register (RESCHn)	3658
Figure 2143. 32-bit V Vector Channel n register (V32_n)	3659
Figure 2144. 64-bit V Vector Least Significant Bytes Channel n register	3660
Figure 2145. 64-bit V Vector Most Significant Bytes Channel n register (V64Mn).	3660
Figure 2146. DMA Source Address Channel n register (DSACHn).	3661

Figure 2147. DMA Destination Address Channel n register (DDACHn)	3661
Figure 2148. DMA Transfer Count Channel n register (DCNTCHn)	3662
Figure 2149. DMA Transfer Control Channel n register (DCTLCHn).	3663
Figure 2150. AMU Status and Configuration register (STCFGn).	3665
Figure 2151. Error Status Register (ESR)	3668
Figure 2152. DMA V Vector Holdoff Count Channel n register (DVHCHn)	3669
Figure 2153. WKPU block diagram	3676
Figure 2154. NMI Status Flag Register (NSR).	3678
Figure 2155. NMI Configuration Register (NCR)	3680
Figure 2156. GTMINT block diagram.	3688
Figure 2157. GTM Module Configuration Register (GTMMCR).	3710
Figure 2158. GTM Interrupts Clear Register (GTMINCLR).	3711
Figure 2159. GTM AEI Control Register (GTMAEICR).	3713
Figure 2160. Emulation and debug device architecture.	3724
Figure 2161. BD clock structure.	3725
Figure 2162. C language encodeECC32_upper function description (Part 1 of 4)	3733
Figure 2163. C language encodeECC32_upper function description (Part 2 of 4)	3734
Figure 2164. C language encodeECC32_upper function description (Part 3 of 4)	3735
Figure 2165. C language encodeECC32_upper function description (Part 4 of 4)	3736
Figure 2166. BD_SIUL2 Device Identification Register 1 (BD_SIUL2_DIDR1).	3740
Figure 2167. BD_SIUL2 Device Identification Register 2 (BD_SIUL2_DIDR2).	3741
Figure 2168. BD_SIUL2 Initialization Status Register (BD_SIUL2_ISR).	3742
Figure 2169. BD_SIUL2 Reset Status Register (BD_SIUL2_RSR).	3743
Figure 2170. DCI block diagram	3746
Figure 2171. DCI in an emulation configuration.	3747
Figure 2172. DCI operating modes state diagram	3749
Figure 2173. DCI JTAG/LFAST switch integration	3750
Figure 2174. DCI switching between JTAG and LFAST modes	3751
Figure 2175. DCI JTAG to LFAST switching diagram	3753
Figure 2176. DCI LFAST to JTAG switching diagram	3754
Figure 2177. DCI debug control mechanism	3757
Figure 2178. DCI debug signal generation	3758
Figure 2179. DCI EVTO management.	3759
Figure 2180. DCI EVTO timing diagram	3760
Figure 2181. DCI control register (DCI_CR)	3761
Figure 2182. JTAG (IEEE 1149.1) block diagram	3766
Figure 2183. 6-bit instruction register	3769
Figure 2184. Device identification register.	3770
Figure 2185. JTAG_PASSWORD register.	3770
Figure 2186. Shifting data through a register.	3771
Figure 2187. IEEE 1149.1-2001 TAP controller finite state machine.	3772
Figure 2188. JTAGM block diagram	3779
Figure 2189. LFAST-to-DCI flow diagram	3780
Figure 2190. JTAGM to DCI serial interface diagram	3783
Figure 2191. Bit ordering for LFAST to JTAGM transfer	3784
Figure 2192. Bit ordering for JTAGM to LFAST transfer	3784
Figure 2193. JTAGM_MCR register	3785
Figure 2194. JTAGM_SR register in LFAST mode	3787
Figure 2195. JTAGM_SR register in SW mode	3788
Figure 2196. JTAGM_DOR0 register.	3790
Figure 2197. JTAGM_DOR1 register.	3790
Figure 2198. JTAGM_DOR2 register.	3791

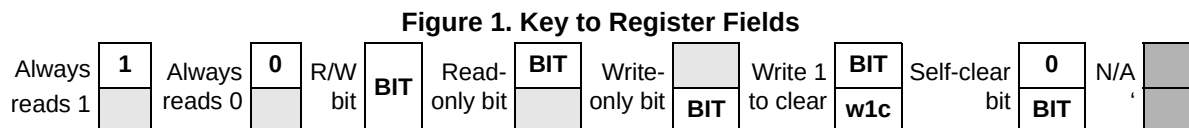
Figure 2199. JTAGM_DOR3 register	3791
Figure 2200. JTAGM_RxCRC register	3792
Figure 2201. JTAGM_DIR0 register	3792
Figure 2202. JTAGM_DIR1 register	3793
Figure 2203. NAR top level block diagram	3794
Figure 2204. NAR control register (NAR_CR)	3797
Figure 2205. NAR status register (NAR_ST)	3799
Figure 2206. Source filter register (NAR_SFR)	3801
Figure 2207. Message type filter register (NAR_TFR)	3802
Figure 2208. Trace memory base address register NAR_TBAHI	3804
Figure 2209. Trace memory base address register NAR_TBALO	3804
Figure 2210. Trace memory bus control register (NAR_TCR)	3805
Figure 2211. Suppress trigger control registers (NAR_STCR1–4)	3806
Figure 2212. Client suppress status register (NAR_CSSR)	3807
Figure 2213. Client disable register (NAR_CDR)	3808
Figure 2214. Nexus Aurora Phy control register (NAR_NAPCR)	3810
Figure 2215. Trace memory fill level and partition configuration register (NAR_AHFPAR)	3811
Figure 2216. AUX bus parsing	3815
Figure 2217. Queue partitioning	3819
Figure 2218. Timestamp architecture	3826
Figure 2219. Nexus controller state machine	3828
Figure 2220. Shifting data into register	3829
Figure 2221. Nexus command format for register selection	3830
Figure 2222. NAL block diagram	3834
Figure 2223. Nexus Aurora Link General Status Register (NAL_GSR)	3837
Figure 2224. NAL General Control Register (NAL_GCR)	3838
Figure 2225. NAL Training Control Register (NAL_TCR)	3839
Figure 2226. High speed Nexus interface block diagram	3842
Figure 2227. Serializer timing diagram	3845
Figure 2228. DWPU block diagram	3846
Figure 2229. Incoming message format	3848
Figure 2230. Non-sync data trace write message format	3849
Figure 2231. Sync data trace write message format	3850
Figure 2232. Non-sync M-Data data trace write message format	3850
Figure 2233. Sync M-Data data trace write message format	3851
Figure 2234. Tag RAM organization example	3852
Figure 2235. Address tag filtering	3853
Figure 2236. Range detect comparators	3854
Figure 2237. DWPU trigger mode	3855
Figure 2238. DWPU message format	3855
Figure 2239. JTAG DR for DWPU register access	3857
Figure 2240. GLB_CTRL register	3861
Figure 2241. SRC_MAP_CTRLn register	3862
Figure 2242. FAR_SA_CFGn register	3863
Figure 2243. FAR_RS_CFGn register	3864
Figure 2244. TRC_CTRL register	3864
Figure 2245. TRC_ADR register	3865
Figure 2246. TRC_DATA register	3866
Figure 2247. ORC_CTRL1 register	3866
Figure 2248. ORC_CTRL2 register	3868
Figure 2249. GLB_STAT register	3869
Figure 2250. ORC_CFGn_ADR register	3870

Figure 2251. ORC_CFGn_DH register	3870
Figure 2252. ORC_CFGn_DL register	3871
Figure 2253. ORC_CFGn_DM register	3871
Figure 2254. Read/Write access control/status (RWCS) register	3873
Figure 2255. Read/write access data (RWD) register	3875
Figure 2256. Read/write access address (RWA) register	3876
Figure 2257. JTAG DR for Nexus register access	3876
Figure 2258. LFAST block diagram	3879
Figure 2259. LFAST frame structure	3881
Figure 2260. Serial frame structure	3882
Figure 2261. Key to Register Fields	3885
Figure 2262. Mode Configuration Register (MCR)	3886
Figure 2263. Speed Control Register (SCR)	3888
Figure 2264. Correlator Control Register (COCR)	3889
Figure 2265. Test Mode Control Register (TMCR)	3890
Figure 2266. Auto Loopback Control Register (ALCR)	3891
Figure 2267. Rate Change Delay Control Register (RCDCR)	3892
Figure 2268. Wakeup Delay Control Register (SLCR)	3893
Figure 2269. Ping Control Register (PICR)	3895
Figure 2270. PLL Control Register (PLLCR)	3896
Figure 2271. LVDS Control Register (LCR)	3898
Figure 2272. Unsolicited Tx Control Register (UNSTCR)	3900
Figure 2273. Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)	3901
Figure 2274. Global Status Register (GSR)	3901
Figure 2275. Data frame Status Register (DFSR)	3903
Figure 2276. Tx Interrupt Status Register (TISR)	3904
Figure 2277. Rx Interrupt Status Register (RISR)	3905
Figure 2278. Rx ICLC Interrupt Status Register (RIISR)	3906
Figure 2279. PLL and LVDS Status Register (PLLLSR)	3908
Figure 2280. Unsolicited Rx Status Register (UNSRSR)	3909
Figure 2281. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)	3910
Figure 2282. Top Level Receive Controller	3913
Figure 2283. Hunt Correlation mode	3914
Figure 2284. High Speed 8 Phase Selection	3914
Figure 2285. High Speed 8 Phase Clock Alignment Example	3915
Figure 2286. High Speed 4 Phase Clock Alignment Example	3915
Figure 2287. Low speed 4 phase clock alignment example	3916
Figure 2288. Samplers, each Sampler has 3 Registers	3918
Figure 2289. Sampler 5 Logic	3919
Figure 2290. Extracting Header and Payload from the 16-bit output from Auto correlation	3920
Figure 2291. Transmit Controller Connections	3922
Figure 2292. Example of Sleep mode	3924
Figure 2293. Rx LoopBack mode	3930
Figure 2294. Rx LVDS LoopBack mode	3931
Figure 2295. Tx LoopBack mode without automatic frame generation	3932
Figure 2296. Automatic Loopback Test status signal timings	3933
Figure 2297. Tx LoopBack mode with automatic frame generation	3934
Figure 2298. Tx LVDS Loopback (external) mode with automatic frame generation	3935
Figure 2299. LFAST module clock domains	3938
Figure 2300. External clock muxing of lfast_sysclk	3939
Figure 2301. External clock muxing of lfast_sysclk of 20/26 MHz	3940
Figure 2302. External clock muxing of lfast_sysclk of 10 MHz	3940

Figure 2303.Slow speed 4 phases generation	3941
Figure 2304.Clock Enables and Clock Paths	3941
Figure 2305.Transmit Clocks Muxing	3943

1 Register conventions

Figure 1 and *Table 1* provide the register conventions used throughout this manual.



Warning: Any register bit described as "Reserved" should not be written. Reserved bits are not intended for customer use.

Table 1. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero
1	Resets to one
—	Undefined at reset
u	Unaffected by reset
[signal_name]	Reset value is determined by polarity of indicated signal

1.1 Acronyms and abbreviations

The [Table 2](#) lists some acronyms and abbreviations used in this document.

Table 2. Acronyms and abbreviations

Term	Meaning
ATX	Analog Test Module
AUTOSAR®	Automotive Open System Architecture
BAF	Boot Assist Flash
CAN	Controller Area Network
CGM	Clock Generation Module
CMU	Clock Monitoring Unit
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
DMAMUX	Direct Memory Access Controller Multiplexer
DSPI	Deserial Serial Peripheral interface
DTS	Development Trigger Semaphore
FCCU	Fault Collection and Control Unit
FEC	Fast Ethernet controller
FLEXRAY	automotive network communications protocol
GPIO	General-purpose I/O
GTM	Generic Timer module
IEEE	Institute for Electrical and Electronics Engineers
IIC	Inter-Integrated Circuit (I ² C)
IMA	Indirect Memory Access
INTC	Interrupt Controller
JDC	JTAG Data Communication
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LFAST	LVDS Fast Asynchronous Serial Transmission
LINFlexD	Local Interconnect Network controller
ME	Mode Entry Module (MC_ME)
MEMU	Memory Error Management Unit
Mux	Multiplex
OSC	Oscillator
PASS	Password and Device Security Module
PCM	Platform Configuration Module
PCU	Power Control Unit

Table 2. Acronyms and abbreviations (continued)

Term	Meaning
PFLASH	Flash controller
PIT	Periodic Interrupt Timer
PMC	Power Management Controller
PRAM	RAM controller
PSI5	Peripheral Sensor Interface
RCOSC	RC Oscillator
RGM	Reset Generation Module
RTL	Register transfer language
Rx	Receive
SD ADC	Sigma Delta Analog to Digital Converter
SAG	Safety Application Guide
SAR ADC	Successive Approximation Register Analog to Digital Converter
SEMA4	Semaphore
SENT	Single Edge Nibble Transmission
SIPI	System Interprocessor Interface
SIUL	System Integration Unit Lite
SMPU	System Memory Protection Unit
SoC	System on Chip
SSCM	System Status and Configuration Module
STCU	Self-Test Control Unit
STM	System Timer module
SWT	Software Watchdog Timer
TSENS	Temperature Sensor
TTCAN	Time Triggered CAN
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter
WKPU	Wakeup Unit
XBAR	Crossbar

1.2 References

In addition to this reference manual, the following documents provide additional information on the operation of the SPC57EM80xx:

- IEEE-ISTO 5001-2003 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture

2 Introduction

2.1 SPC57EM80xx microcontroller

The SPC57EM80xx microcontroller is the first in a new family of devices superseding the SPC56xx family. The SPC57EM80xx builds on the legacy of the SPC56xx family, while introducing new features coupled with higher throughput to provide a substantial reduction in cost per feature as well as significant improvements in power and performance (MIPS per mW). There are five processor cores on the SPC57EM80xx device.

Table 3. Processor cores

Computational Shell		
Core0	Main Core_0	e200z420
Core1	Main Core_1	e200z420
Checker Core0	Checker Core_0s	e200z419
I/O Processor		
Core2	Peripheral Core_2	e200z425
Hardware Security Module		
e200z0 Core		

2.1.1 Target applications

This device is targeted at automotive powertrain controller and chassis control applications for six to eight cylinder gasoline and diesel engines, transmission control, steering and braking applications, as well as high-end hybrid and advanced combustion systems.

Many of these applications are considered to be functionally safe and are designed to achieve ISO 26262 ASIL D compliance. For this reason, safety measures have been added which are further described in [Chapter 6: Functional Safety](#), as well as the *SPC57EM80xx Safety Manual*.

2.1.2 Overall architecture

The architecture is split into two distinct regions, as shown in [Figure 2](#): the computational shell and the I/O subsystem. The computational shell consists of three CPUs (two in lock-step for safety), fast local memories inside the CPUs, and fast system memory. All components of the computational shell are connected through a fast crossbar switch (XBAR_0).

The I/O subsystems consists of an I/O processor (IOP), DMA, Hardware Security Module (HSM), and all the other peripherals connected through a slow crossbar switch (XBAR_1).

The computational shell is optimized for performance and the I/O subsystem is optimized for low power.

The two crossbars (XBAR_0 and XBAR_1) are connected by a pair of cross-connected master - slave ports. The result of this is that all crossbar masters on both crossbars, can directly access all crossbar slaves on both crossbars. From a software perspective, every

bus master (all CPUs, DMA, SIPI, Ethernet, FlexRay, HSM) sees every memory and peripheral (all RAMs, flash, and peripherals) in a consistent flat memory map.

2.1.3 Core features

The main computational shell consists of dual e200z4 CPUs operating at up to 200 MHz with a third, identical CPU running as a safety checker core in delayed lockstep mode with one of the dual z4 cores. The I/O subsystem includes a CPU targeted at managing the peripherals. The I/O subsystem core is also an e200z4 CPU running at up to 200 MHz and including enhancements to implement Digital Signal Processing (DSP) algorithms. The fifth programmable CPU is an e200z0 running at up to 100 MHz and embedded in the HSM. All CPUs are compatible with the Power Architecture® VLE instruction set, which supports code size reduction. The Power Architecture has enhancements that improve the architecture's fit in embedded applications.

2.1.4 I/O Processor

The I/O Processor (IOP), also referred to as Peripheral Core_2, is one of five processor cores on the SPC57EM80xx device as shown in [Figure 2](#). In general, the intended use of the IOP is to initially configure the machine and then control various peripheral elements of the device. However, the IOP is not limited to any particular duty.

Note: The e200z0 core is embedded in the Hardware Security Module.

The data bus structure of the IOP, which consists of a 32-bit address bus and a 64-bit data bus, is connected to the slow crossbar switch, which operates at 100 MHz. The instruction bus structure of the IOP, which consists of a 32-bit address bus and a 64-bit data bus, is connected to the fast crossbar switch, which operates at 200 MHz. The address bus structure and the data bus structure of Main Core_0 and Main Core_1 are also connected to the fast crossbar switch. The data bus structure of the IOP is connected to the slow crossbar switch, which operates at 100 MHz. Thus, data fetches run at only one-half the speed of instruction fetches.

It is possible to fetch data from memory elements connected to the fast crossbar switch and it is also possible to fetch instructions from memory elements connected to the slow crossbar switch. In both cases, the instruction and data paths would have to go through both the slow and fast crossbar switches and would be relatively slow. When the IOP first starts, it will start executing from memory elements on the fast crossbar switch. However, good programming techniques will cause the IOP to fill its internal instruction cache so that accesses through the fast crossbar switch occur less often.

The I/O Processor is automatically started during the boot-up sequence after the release of reset. This process is fully described in [Chapter 8: Reset and Boot](#). During the boot sequence the System Status and Configuration Module (SSCM) reads and interprets a set of device configuration records located in the TEST flash area. One of these records is the start address of the code where the IOP will begin program execution. This vector will always point to a location in the Boot Assist Flash (BAF). Once the SSCM finds and interprets this particular device configuration record, it will write the start address for the IOP to the appropriate location in the Mode Entry Module (MC_ME). When the internal reset signal is released, the MC_ME transfers the IOP start address to the IOP and code execution begins at that address.

The code for the Boot Assist Flash, the TEST Device Configuration records and some of the UTEST Device Configuration records are written by the device manufacturer and are programmed into the device during production testing. All SPC57EM80xx devices are

shipped with startup code programmed into the BAF. Also, the SPC57EM80xx is configured such that code execution by the IOP must begin with a start address located in the BAF.

If there is no user application code for the IOP to execute, the IOP, executing preprogrammed code in the BAF, will attempt to download user application code through the LINFlexD port using a specific data transfer protocol. This downloaded code is installed into program memory and code execution will begin at the start address embedded in the downloaded code.

As previously stated, the IOP's data bus structure is connected to the slow crossbar switch, which provides access to many of the peripheral devices on the device. The instruction bus of the IOP is connected to the fast crossbar switch, which gives the IOP access to the BAF, as well as all of the other memory elements. When the IOP begins code execution, a considerable amount of traffic through the fast crossbar switch will be required. The IOP has internal instruction cache. Once the IOP gets started and the internal instruction cache begins to fill, significantly less traffic will be required by the IOP through the fast crossbar switch. Accesses to the peripheral elements on the slow crossbar switch is transparent to activity occurring on the fast crossbar switch.

The IOP is placed in the SPC57EM80xx architecture in such a way that enables it to access many of the peripheral elements without affecting the processing speed of Main Core_0 and Main Core_1. On the SPC57EM80xx device, there are several analog-to-digital channels taking data that must be analyzed for various spectral components.

The IOP has an integrated DSP processor which has special hardware to execute multiply and accumulate operations. Because the DSP unit functions as a coprocessor integrated into the e200z425 core, data fetched by the z425 core can be evaluated with much greater speed than if the data had to reside in a memory element outside the z425 core.

One intended purpose for the IOP is to manage a system of analog-to-digital converters and fetch the digital data that is generated. The integrated DSP unit uses a signal processing algorithm to digest the incoming data. The results of this signal processing algorithm are used in other engine control algorithms. Using the IOP to manage the analog-to-digital converters and do the initial signal processing of the data allows Main Core_0 and Main Core_1 to focus on other compute intensive applications instead of managing relatively slow peripheral elements.

Once the IOP has performed its initialization duties (as determined by preloaded instruction code in the BAF) and downloaded any needed applications code, the IOP can essentially be programmed to perform any duty or access any memory element. When executing instructions from its internal instruction cache and fetching data from peripherals connected to the slow crossbar switch, the IOP operates autonomously from Main Core_0 and Main Core_1.

2.1.5 Memory hierarchy

The SPC57EM80xx computational shell has three levels of memory hierarchy. At the top level is a combination of instruction cache, data cache, local instruction RAM, and local data RAM, depending on the specific CPU. The local RAM arrays are embedded into the CPU's pipeline to provide the fastest possible access time. The second level of memory is system RAM and flash memory. These are connected by a 200 MHz, 64-bit wide crossbar. The third level of memory system is the I/O subsystem. All of the I/O peripherals are connected together by a 100 MHz, 64-bit wide crossbar switch, while peripherals are connected at either 100 or 50 MHz, depending on their need. The reduced speed for the peripherals helps to reduce power.

2.2 Features summary

On-chip modules within SPC57EM80xx include the following features:

- Three main CPUs, 32-bit CPU core complexes (e200z4), two of which run in lockstep
 - Power Architecture embedded specification compliance
 - Instruction set enhancement allowing variable length encoding (VLE), encoding a mix of 16-bit and 32-bit instructions, for code size footprint reduction
 - Single-precision floating point operations
 - 16 KB Local instruction RAM and 64 KB local data RAM
 - 8 KB I-Cache and 4 KB D-Cache
- I/O Processor, 32-bit CPU core complexes (e200z4), with:
 - Power Architecture embedded specification compliance
 - Instruction set enhancement allowing variable length encoding (VLE), encoding a mix of 16-bit and 32-bit instructions, for code size footprint reduction
 - Single-precision floating point operations
 - Lightweight Signal Processing Auxiliary Processing Unit (LSP APU) instruction support for digital signal processing (DSP)
 - 16 KB Local instruction RAM and 64 KB local data RAM
 - 8 KB I-Cache
- 4304 KB on-chip flash (4000 KB code flash, 288 KB data flash, and 16 KB UTest flash: supports read during program and erase operations, and multiple blocks allowing EEPROM emulation)
- Up to 450 KB of total on-chip SRAM (128 KB general purpose SRAM, 192 KB data RAM, 48 KB instruction RAM, 35 KB GTM RAM, 16 KB overlay RAM and 32 KB peripheral RAM)
- 64-channel direct memory access controller (eDMA)
- 4 Interrupt controllers (INTC) (one INTC per user programmable CPU)
- Dual Phase-locked loops with stable clock domain for peripherals and FM modulation domain for computational shell
- Dual Crossbar switch architecture for concurrent access to peripherals, flash, or RAM from multiple bus masters with end-to-end ECC
- Hardware Security Module (HSM) to provide robust integrity checking of flash memory
- System Integration Unit Lite (SIUL2)
- Boot Assist Module (BAM) supports factory programming using serial bootloader through 'UART Serial Boot Mode Protocol'. Physical interface (PHY) can be:
 - UART/LIN
 - CAN
 - FlexRay
- GTM103 – generic timer module
 - Intelligent complex timer module
 - 120 channels (32 input and 88 output)
 - 4 programmable fine grain multi-threaded cores
 - 34 KB of RAM
 - 24-bit wide channels

- Hardware support for engine control, motor control and safety related applications
- Enhanced analog-to-digital converter system with
 - Eight separate 12-bit SAR analog converters
 - Six separate 16-bit Sigma-Delta analog converters
- Seven deserial serial peripheral interface (DSPI) modules
- Five LIN and UART communication interface (LINFlexD) modules
 - LINFlexD_0 is a Master/Slave
 - LINFlexD_1, LINFlexD_2, LINFlexD_14 and LINFlexD_15 are Masters
- Three modular controller area network (M_CAN) modules, and one time-triggered controller area network (M_TTCAN)
- Dual-channel FlexRay controller
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2010 standard.
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1 and 1149.7)

2.3 Feature list

[Table 4](#) lists a summary of major features for the SPC57EM80xx device. The feature column represents a combination of module names and capabilities of certain modules. A detailed description of the functionality provided by each on chip module is given later in this document.

Table 4. Features List

Feature	Description
SPC57xxfamily	55 nm
Main Cores	
Main Core	e200z4
Number of Main Cores	2
Number of checker cores	1
Local RAM (per main core)	16 KB Instruction 64 KB Data
Single Precision Floating Point	Yes
VLE	Yes
Cache	8 KB Instruction 4 KB Data
Peripheral Core	
Peripheral Core	e200z4
Local RAM	16 KB Instruction 64 KB Data
Single Precision Floating Point	Yes
SIMD	Yes

Table 4. Features List (continued)

Feature	Description
VLE	Yes
Cache	8 KB Instruction
Other	
MMU Entries	0
MPU	Yes
Semaphores	Yes
CRC Channels	2
Software Watchdog Timer (SWaT)	4
Core Nexus Class	3+
Sequence Processing Unit (SPU)	16
Debug and Calibration Interface (DCI) / Run control Module	Yes
System SRAM	128 KB
Flash	Code Flash: 4000 KB Data Flash: 288 KB UTest Flash: 16 KB
Flash fetch accelerator	Two 64-bit AHB ports, 256-bit read data interface. Each AHB port contains 4-entry, 2-way set-associative mini-cache.
Flash Overlay RAM	16 KB
Calibration Interface	64-bit IPS Slave
DMA channels	64
DMA Nexus Class	3
LINFlexD	5
M_CAN	3
M_TTCAN	1
DSPI	7
Microsecond channel downlink	Yes
SENT bus	10
I ² C	1
PSI5 bus	3
FlexRay	1 × Dual channel
Ethernet	Yes
SIPI / LFAST Interprocessor bus	High-Speed
System Timers	8 PIT chan 4 AUTOSAR [®] (STM)
GTM Timer	32 Input Channels, 88 Output Channels

Table 4. Features List (continued)

Feature	Description
GTM RAM	34 KB
Interrupt controller	446 sources
ADC (SAR)	8
ADC (SD)	6
Temp. sensor	Yes
PLL	Dual PLL with FM
External Power Supplies	5 V, 3.3 V, 1.2 V
Low Power Modes	Stop Mode Slow Mode

2.4 Packaging

The SPC57EM80xx is available in the following production packages: eLQFP176 and LFBGA292. The SPC57EM80xx is also available in a 256-pin Fusion Quad package for emulation and debug devices, which provides a eLQFP176 pin-compatible package with additional signals for development. The LFBGA292 package supports production and emulation and debug devices with compatibility for an emulation package in a production device footprint.

2.5 Software debug and calibration

The Production Device (PD) includes many features to facilitate software development and calibration of constant data. These features include:

- Nexus code execution trace on all CPUs
- Nexus data trace on all bus masters (DMA, Interprocessor bus, CPUs, and so on)
- Performance monitors for each CPU
- Sequence Processing Unit for complex trace triggering
- 16 KB of overlay RAM

For more complex software development and calibration of constant data, a Buddy Device (BD) is packaged together with the SPC57EM80xx production device as a multi-die emulation and is referred to as an Emulation and Debug Device (ED).

The ED provides exactly the same features as the PD. The ED is available in the same package as the PD but has an additional overlay RAM and a high-speed Nexus trace output port consisting of four lanes of Aurora (LVDS) operating up to 1.25 Gbit/s.

Block diagrams

The figures below show the top-level block diagrams.

Figure 2. Block diagram

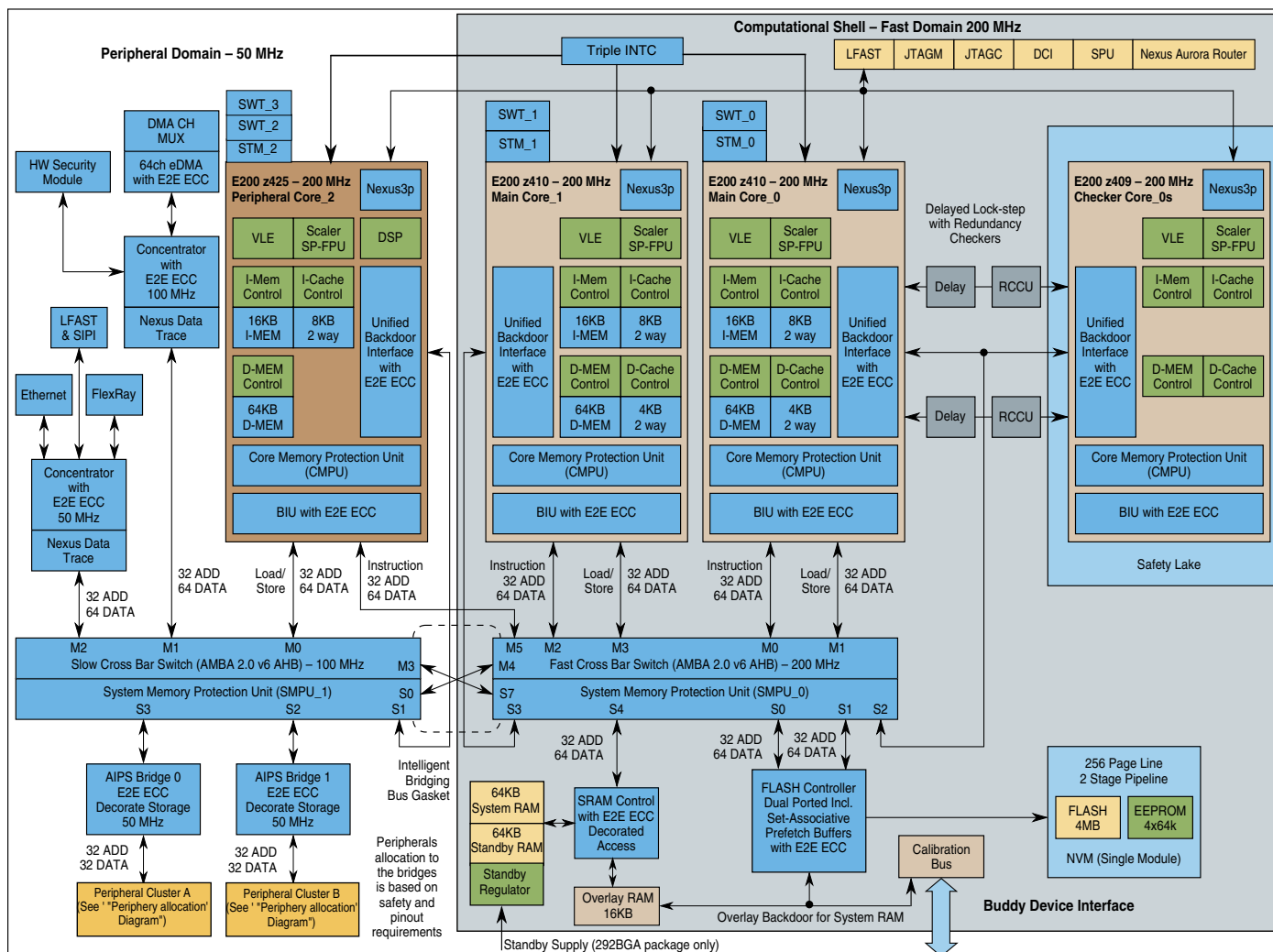
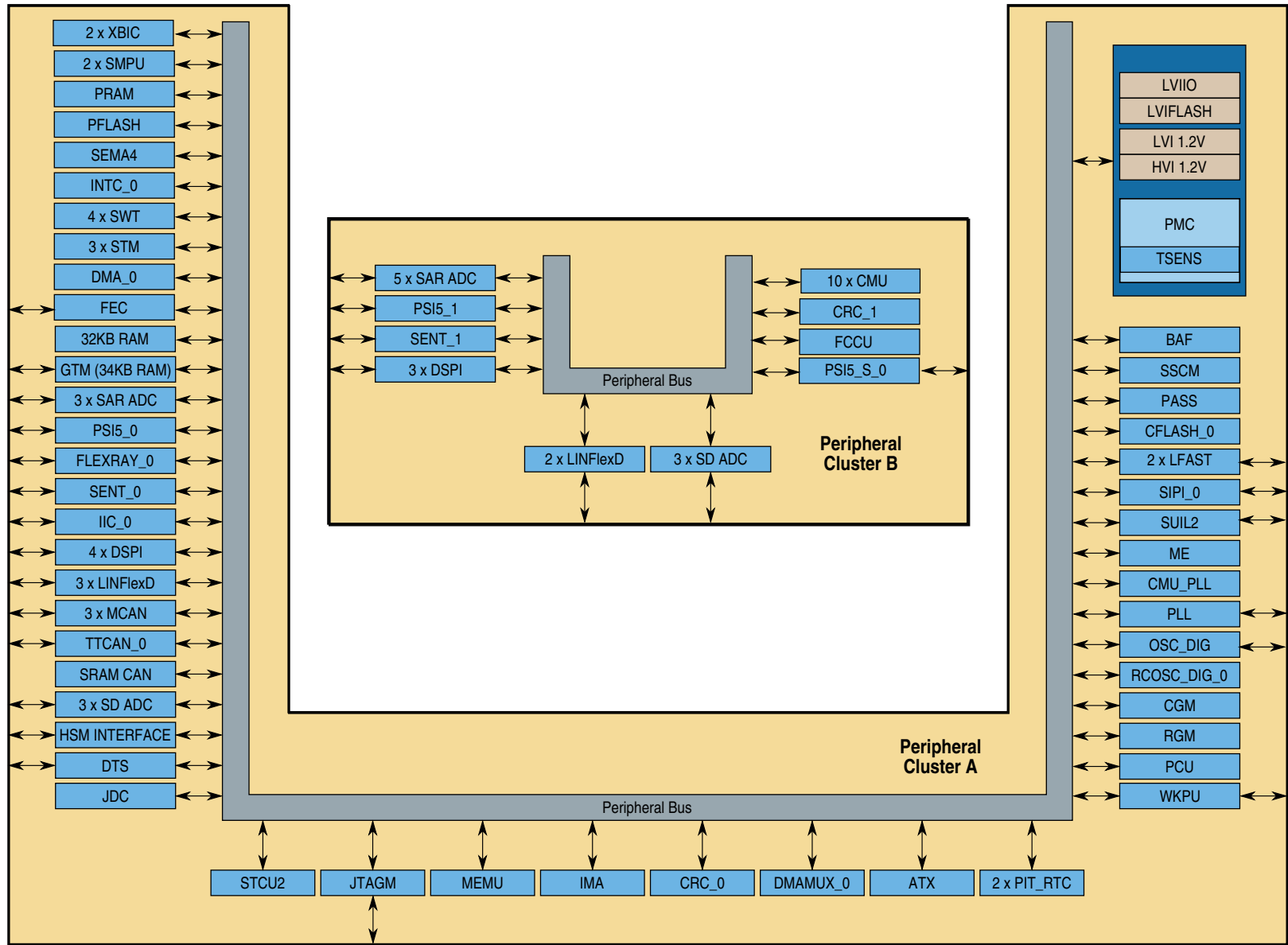


Figure 3. Periphery allocation



3 Embedded memories

3.1 Overview

The embedded memory architecture for SPC57EM80xx devices is a significant departure from previous-generation devices. Features include:

- Onboard SRAM, including system SRAM, local instruction and data memory for each processor core, and overlay SRAM
- Onboard system error correction code (ECC) flash memory
- End-to-end ECC (e2eECC) error detection and correction
- Built-in flash memory security features including censorship and a programmable Hardware Security Module (HSM)
- Embedded memories in the peripherals^(a):
 - AMU
 - CAN
 - eDMA
 - FEC
 - FlexRay
 - GTM

3.2 SRAM

3.2.1 System SRAM

SPC57EM80xx devices include 128 KB of general-purpose on-chip ECC SRAM, including 64 KB standby RAM. The SRAM can be configured for either zero- or one-wait state read operation latency using the PFCR1 register in the SRAM controller.

- See [Chapter 33: RAM controller \(PRAM\)](#) for details on configuring the SRAM controller.
- See [Table 15](#) in [Chapter 5: Memory Map](#) for the SPC57EM80xx SRAM map.

3.2.2 Standby SRAM Regulator

A total of 64 K of the system SRAM on the device can remain powered when the rest of the device is powered down. This is achieved with the dedicated VDDSTBY pin in the packages. When the standby RAM is powered by VDDSTBY, the RAM is not operational, but the contents are retained.

a. Refer to the respective module chapter for details.

The Standby RAM regulator has the following key features:

- Automatic, safe, switch between operational (VDD_LV) and standby (VDDSTBY) supplies
- Internal regulation of the high voltage VDDSTBY supply to the low voltage standby RAM supply
- Brownout detection (minimum RAM data retention voltage violation) and reporting via a S/W clearable register bit

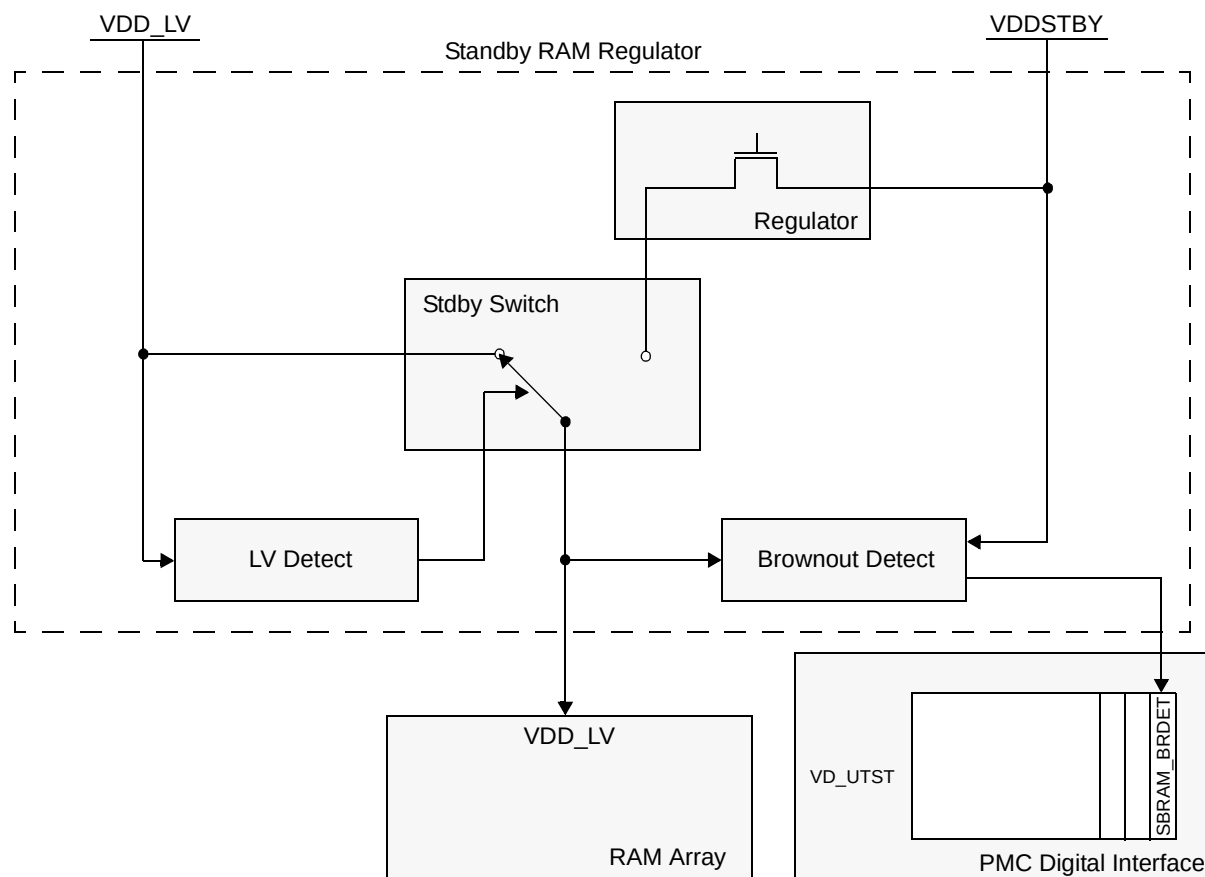
The Standby RAM regulator block diagram is given in [Table 4](#).

There are two primary modes of operation with the standby RAM feature:

- Standby RAM feature is not required in the application - in this case, the VDDSTBY pin is connected to ground, and the standby RAM on the device is supplied by VDD_LV and will behave as normal RAM for all conditions.
- Standby RAM feature is required in the application - in this case, the VDDSTBY pin is connected to a voltage within the range given in the device data sheet. The VDDSTBY pin is assumed to be connected to a constant supply, and present before and after the power up/down of the device LV and HV supplies. When the VDD_LV supply to the RAM is above the LV operational voltage threshold, the standby RAM is fully operational and behaves exactly the same as the non-standby system RAM. When the VDD_LV supply falls below the LV Detect threshold, the RAM low voltage supply is switched to the standby supply. For voltages above 1.2V, the standby regulator is enabled. For voltages below 1.2V, the VDDSTBY pin voltage is passed through directly to the RAM. See the device data sheet for min/max VDDSTBY values, LV supply thresholds, and the operational and brownout detection voltages.

The brownout detection feature of the standby RAM regulator informs the user if any voltage dip caused a RAM retention failure in standby mode. The brownout detector monitors the supply to the RAM array, and sets a latch in the PMC digital interface, when the voltage falls below the minimum RAM retention voltage given in the device data sheet. The latch is set on power up of the device. S/W then clears the latch by writing a '1' to the SBRAM_BRDET bit in the VD_UTST register of the PMC digital interface. If the minimum RAM retention voltage is violated, the latch will be set again. The latch is set on every power-up of VDD_LV unless VDDSTBY is present and within the specified ranges in the data sheet. See chapter [Section 60.6: Standby RAM Regulator Interface Logic](#), for more information on the VDDSTBY pin and the standby RAM regulator.

Figure 4. Standby RAM Regulator Block Diagram



3.2.3 Overlay SRAM

Overlay SRAM is included in the SPC57EM80xx family of devices as part of a comprehensive set of calibration and debug features. Overlay SRAM can be mapped over specific regions of on-chip flash memory so that any access to an overlaid flash address is routed to the overlay SRAM instead. This enables the calibration of constant data without requiring additional external RAMs and calibration memory interfaces. The SPC57EM80xx has two 8 KB banks of ECC-protected RAM, for a total of 16 KB.

Note: *Overlay SRAM is normally used for system development purposes only. Code can be executed from internal overlay RAM and extended overlay RAM where available. Execution from overlay RAM remapped over Flash can be enabled or disabled by use of PFCRCR[IRMEM]. Code can be executed directly from Extended Overlay RAM (address range beginning 0x0C000000) on the ED device or internal Overlay RAM (address range beginning 0x0D000000) regardless of the PFCRCR[IRMEM] setting.*

Overlay memory can also be used to hold the device debug trace stream without requiring complex debug tool hardware and access to a high-speed trace port. Timing for calibration accesses to a particular overlay SRAM is the same as access to the underlying flash memory if the overlay SRAM is not being used as a destination for trace streaming at the same time.

The calibration remap function supports three different types of overlay SRAM:

- An internal overlay RAM is included on all standard production devices.
- An extended overlay RAM is an additional memory that is available only on special emulation and debug devices.
- A portion of the system RAM can be used as the overlay RAM.

The overlay function is implemented using registers in the flash memory controller.

1. Using PFlash Calibration Region Descriptors (PFCRDn) define one or more (up to 32) overlay regions.
2. Enable individual regions using the appropriate bits in the PFLASH Remap Descriptor Enable Register (PFDCRDE).
3. Enable overlay using the PFCRCR[GRMEN] field.

See [Section 34.5.14: PFlash calibration remap support](#), for details on memory overlay (remapping).

3.2.4 Processor core local SRAM

SPC57EM80xx devices include local instruction and data SRAM (I-MEM and D-MEM) for each processor core to provide enhanced performance.

- Each main processor core has 16 KB of instruction SRAM and 64 KB of data SRAM.
- The peripheral processor core has 16 KB of instruction SRAM and 64 KB data SRAM.

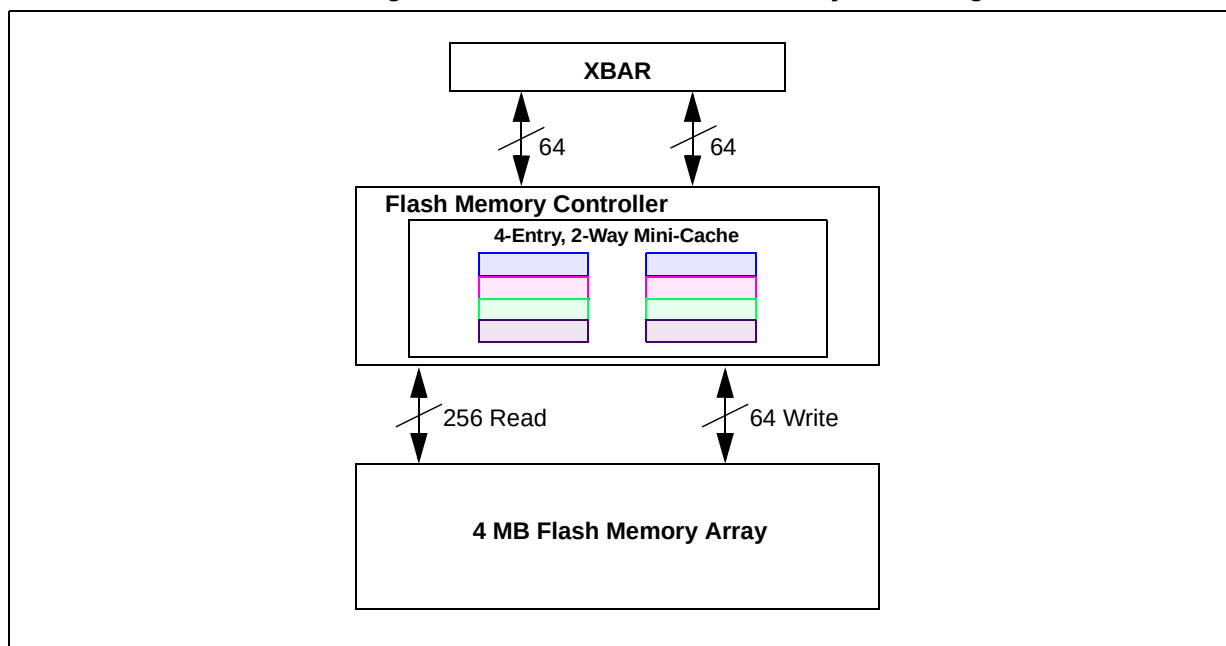
The core SRAM supports zero wait-state, single-cycle latency on processor local accesses.

Each area of core SRAM is accessible by other processor cores and bus mastering peripheral devices. See [Table 15](#) in [Chapter 5: Memory Map](#) for details.

3.3 Embedded flash memory

Flash memory on SPC57EM80xx devices consists of a flash memory controller and a flash memory array module. The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the device crossbar switch. [Figure 5](#) shows the memory architecture.

Figure 5. SPC57EM80xx flash memory block diagram



The remaining sections give the functional details of the flash controller and flash array, followed by the memory maps.

3.3.1 Flash memory controller

The SPC57EM80xx flash memory controller acts as an interface between the system bus and the flash memory array and serves as the interface to the on-chip overlay RAM and off-chip Buddy Device.^(b)

The flash memory controller contains a four-entry, two-way set-associative mini-cache that delivers flash read data with a zero-wait state response on lines that reside in the cache. Each entry contains one flash page, a 256-bit (32-byte) memory value. Read requests that miss the cache generate the needed flash memory array access.

The flash memory controller contains configuration registers that manage flash functionality such as read buffering in the mini-cache, access control, calibration RAM overlay remapping, and read wait state management of the flash memory.

See [Chapter 33: RAM controller \(PRAM\)](#), for details.

3.3.2 Flash memory array

SPC57EM80xx devices include a single 4 MB flash memory array, referred to as “main space,” plus an independent 16 KB block of one-time-programmable (OTP) flash memory included to support systems that require non-volatile memory for security features or system initialization information. The independent 16 KB block is referred to as “UTEST space.”

b. The Buddy Device is an optional companion chip with additional memory and debug features. A Buddy Device is combined with a Production Device to form an Emulation and Debug Device (ED) for use during system development. It is not used in production systems.

Reads of the embedded flash memory return a 256-bit page of data that may be buffered in the mini-cache in the flash memory controller. Programming of the flash may be done by double word (64-bits), page (256-bits), or quad-page (1024-bits). Flash memory is erased on a block by block basis.

3.3.2.1 Features

- Flash segmentation
 - 16 KB, 32 KB, 64 KB, and 256 KB blocks provided for code or data
 - Four 64 KB blocks provided for EEPROM emulation
 - One 16 KB block and two 64 KB block provided for secure code
 - Two 16 KB blocks provided for secure data
 - Support for reading-while-writing when the accesses are to different partitions
- Flash memory protection: write protection and OTP function available for each block
- Test and initialization data stored in a non-volatile 16 KB OTP UTEST block
- ECC with double bit detection, single bit correction
- Erase suspend, program suspend, and erase-suspended program all supported

3.3.2.2 Flash organization

The embedded flash memory consists of six address spaces in four groupings:

- 16 KB low address space, 32 KB low address space, and 64 KB low address space
- 16 KB mid address space
- 64 KB high address space
- 256 KB address space

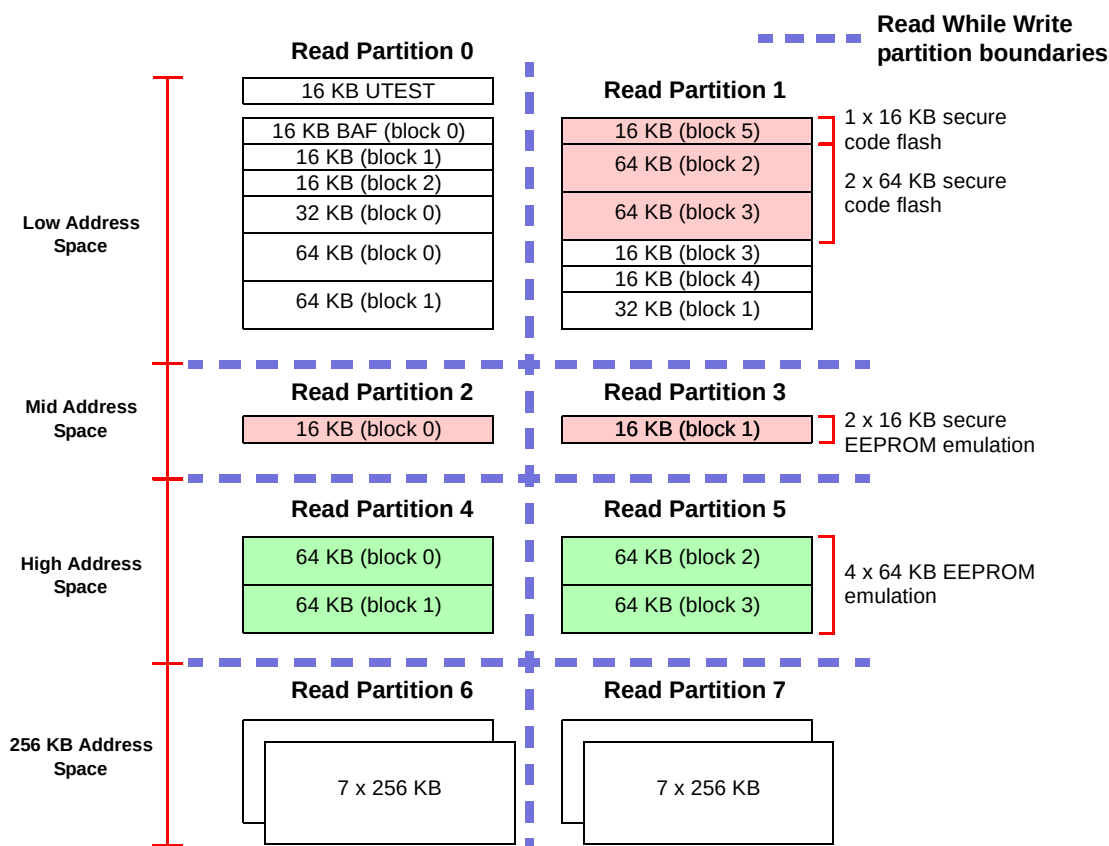
Address space group (low-, mid-, or high-address space) determines which registers/fields are used to select blocks for modify operations or lock them from modify operations. Each address space is independent.

The flash module is further divided into eight partitions that determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a “write” (program or erase) to a given partition, it can simultaneously read from any other partition.

- For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written).
- For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

[Figure 6](#) shows the flash block segmentation and read-while-write partitioning for SPC57EM80xx devices.

Figure 6. Flash memory segmentation and read-while-write partitioning



Note: See [Table 13](#) in [Chapter 5: Memory Map](#) for the SPC57EM80xx flash memory map.

3.3.2.3 UTEST memory space

Devices in the SPC57EM80xx family contain a 16 KB area of One-Time Programmable (OTP) flash memory for storing test information and device configuration data. See [Table 16](#) in [Chapter 5: Memory Map](#) for the SPC57EM80xx UTEST flash memory map.

3.4 End-to-end Error Correction Code (e2eECC)

To support market requirements related to improved functional and transient fault detection capabilities, this family of automotive microcontrollers includes end-to-end ECC (e2eECC) support. This e2eECC is structurally different to traditional “ECC at memory” functionality because it provides robust error detection capabilities from one endpoint of an information transfer to another endpoint, with temporary information storage in one or more intermediate components.

Memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect and/or correct errors that have occurred on data stored in the memory. On the other hand, e2eECC generates error protection codes at the source of data generation. It sends the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master and performs a data integrity check using the previously stored error protection codes at a data memory when a

read of stored information is requested by a bus master. The intermediate storage may transform the generated error protection codes into another format for storage and then regenerate the codes for provision when a request is made to read the stored information or it may simply store the original protection codes unaltered, depending on the particular unit.

Additionally, the error protection codes are generated on the basis of more than just the data associated with a storage location to protect additional information associated with an access. In particular, address information corresponding to the access location of the stored information is combined with the stored data at the data source to generate error protection codes that cover certain types of addressing errors in the system interconnect or in the memory unit. The error protection codes may be checked at the memory unit on a store to ensure that no address information or data has been corrupted while the request has transitioned through the device from the bus master source. The error protection codes may also be checked to ensure that address decoding within the storage memory was performed properly (although not all address decoding errors can be detected this way) or it may simply store the received data and protection codes at the address it receives.

On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location or locations and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory or in the retrieved data. This ensures that the address sent for the request was not corrupted, the addressed location was actually accessed (to the extent it is possible to ensure) and the stored data was error free, to the extent the ECC coding scheme is able to detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust than the previous implementation of locally generated and checked/corrected error protection at each memory unit.

A comparison of the traditional and e2eECC approaches is shown in [Table 5](#).

Table 5. ECC comparison of data write-then-read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data and checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave

Table 5. ECC comparison of data write-then-read sequence (continued)

Traditional ECC	End-to-End ECC (e2eECC)
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

The scope of differences in the operations “covered” by the ECC checks is readily apparent with the e2eECC concept providing improved fault detection capabilities in two important aspects:

- The entire data transaction; from the initiating bus master, through the entire platform crossbar steering mechanism, to the destination slave target is covered during write accesses. Likewise, a read is checked from the initiating bus master, through the crossbar steering mechanism, through the actual memory read and transmission of the data plus checkbits back to the bus master, where the integrity and correctness of the entire transaction is checked.
- The selected ECC provides protection of both the address field as well as the data field, again for improved fault coverage.

Additionally, this particular structure also provides a significant implementation improvement. The traditional ECC at the memory approach places the checkbit decode and error syndrome generation with the error/no_error state, affecting whether the system bus transfer must be stalled (to correct a single bit error or report a noncorrectable event) or is allowed to complete (for error-free transfers) in a critical timing arc which often sets the upper limit of the operating frequency of the microcontroller. With the e2eECC scheme, the checkbit decode and error syndrome logic is located in the requesting bus master where there are generally more degrees of implementation freedom and the error/no_error state determination is removed from the system bus cycle termination logic; producing an improved timing arc and generally higher operating speeds.

Errors that occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in corrupted read data or checkbits being returned to a requester. While not all possible errors can be detected this way, this approach provides a substantial improvement versus traditional methods. Additional on-line diagnostics, or additional hardware, or both should be able to catch a majority of the errors not covered directly by the e2eECC scheme.

3.5 Security features

The SPC57EM80xx architecture uses a combination of device life cycle monitoring and password protection to limit the conditions under which debugging interfaces can access certain areas of flash memory.

Note: Detailed information on the SPC57EM80xx security architecture is published in a separate document and is available on a limited basis to customers who have a non-disclosure agreement (NDA) with STMicroelectronics.

4 Signal Description

4.1 Production packages

The following production packages are for the SPC57EM80xx device:

- eLQFP176 Exposed Pad
- LFBGA292

The BGA292 is a superset of the eLQFP176 pins. The LFBGA292 adds only:

- Additional GPIO
- More analog inputs
- Added muxing options for the eLQFP176 I/O

There are no I/O interfaces defined for the LFBGA292 that are not included in the eLQFP176 (external bus, etc.).

Case number and outline drawings for each package are provided in the *SPC57EM80xx Microcontroller Data Sheet*.

4.2 Emulation package

A 216-pin FusionQuad QFP package is used for application debug and calibration support with the LQFP package for the SPC57EM80xx. The emulation and debug device also comes in a LFBGA292 package. This package has 20 pins on the inner top/bottom rows that allow the ED signals to be placed there, and therefore not affect the outer row pinout. These ED signals are connected from the inner row pins of the FusionQuad package to bond sites in the pad ring of the production device. Those bond sites are connected to the emulation and debug device I/O pads with routing on the die.

4.3 Package pinouts and ballouts

For pin package pinouts and ballouts refer to the SPC57EM80xx Microcontroller Data Sheet.

4.4 Pin/ball descriptions

The complete Signal Description of the SPC57EM80xx device is provided the following sections:

- [Section 4.4.1, Pin/ball startup and reset states](#)
- [Section 4.4.2, Power supply and reference voltage pins/balls](#)
- [Section 4.4.3, System pins/balls](#)
- [Section 4.4.4, LVDS pins/balls](#)
- [Section 4.4.5, Generic pins/balls](#)

The signal multiplexing options appear in the I/O Signal Description and Input Multiplexing Table^(c)

4.4.1 Pin/ball startup and reset states

This table provides startup state and reset state information for device pins/balls.

Table 6. Pin/ball startup and reset states

Pin	Startup state ⁽¹⁾	State during reset	State after reset
I/O pins ⁽²⁾	Weak pullup	Input, weak pullup	Input, weak pullup
Analog Inputs ⁽³⁾	High-impedance	High-impedance	High-impedance
F0 (ERROR0, FIO) ⁽⁴⁾	High-impedance	Input, high-impedance	Input, high-impedance
ESR0	Output	Output, strong pulldown	Input, weak pullup
ESR1	Weak pullup	Input, weak pullup	Input, weak pullup
PORST	Weak pulldown	Input, weak pulldown	Input, weak pulldown
TESTMODE ⁽⁵⁾	Weak pulldown	Input, weak pulldown	Input, weak pulldown
JCOMP (TRST)	High-impedance	Input, weak pulldown	Input, weak pulldown
TDI	High-impedance	Input, weak pullup	Input, weak pullup
TDO	High-impedance	Weak pullup	High-impedance
TMS	High-impedance	Input, weak pullup	Input, weak pullup
TCK	High-impedance	Input, weak pulldown	Input, weak pulldown
XTAL/EXTAL ⁽⁶⁾	High-impedance	High-impedance	High-impedance

1. Startup state is exited when the core and high-voltage supplies reach minimum levels as defined in the Power Management chapter.
2. Pins on the device that are not supply or reference are not explicitly defined in this table.
3. Analog input without output function—weak pull-up/weak pull-down can be activated only after reset sequence completion
4. F0 is also referred to as ERROR0 and FIO.
5. The SPC57EM80xx MCU does not implement a system configuration pin—TESTMODE is the only pin that affects device configuration. It is latched on the PORST rising edge and internally pulled-down.
6. Function of XTAL/EXTAL pins defined by the enable bit in the flash user test row. If enabled, the pins are analog inputs to the oscillator. If disabled, the XTAL/EXTAL pins default to TBD.

4.4.2 Power supply and reference voltage pins/balls

[Table 7](#) contains information on power supply and reference pin functions for the devices.

- c. The I/O Signal Description and Input Multiplexing Tables is located in a Microsoft Excel[®] workbook attached to this document.

Table 7. Power supply and reference pins

Supply			QFP pin		BGA ball	
Symbol	Type	Description	176	FQ216	292PD	292ED
V _{SS_HV}	Ground	High voltage ground	—	A1, A4, A8, A11, A14, A17, A20, A26, A29, A40	A1, B2, D4, E5, A20, B19, D17, E16, T16, U17, W19, Y20	
V _{SS_LV}	Ground	Low voltage ground	—	—	P12, P9, N9-N12, M7, M8, M10, M11, M13, M14, L8-L13, K8-K13, J13, J14, J10, J11, J7, J8, H9-H12, G12, G9	
V _{SS_HV_OSC}	Ground	Ground supply for the oscillator	101		L20	
V _{SS_HV_ADV}	Ground	Ground supply for ADC	53		Y4	
V _{DD_LV}	Power	Low voltage power supply for production device (PLL is also powered by this pin.)	24, 68, 100, 123		P13, P8, N14, N7, H14, G13	
V _{DDSTBY}	Power	Standby RAM Supply	—		T11	
V _{DD_LV_BD}	Power	Low voltage power supply for buddy die	10	10, A12	—	H7, G8
V _{DD_HV_PMC}	Power	High voltage power supply for internal power management unit	153		—	
V _{DD_HV_PMC_BYP}	Power		—		A19, B18	
V _{DD_HV_IO_MAIN}	Power	High voltage power supply for I/O	25, 69, 99, 139, 171	25, 69, 99, 139, 171, A23, A30	W20, V19, B3, A2	
V _{DD_HV_IO_BD}	Power	High voltage power supply for buddy die I/O	—	A7	—	K14
V _{DD_HV_IO_JTAG}	Power	Oscillator and JTAG pin supply	104		N20	
V _{DD_HV_IO_FLEX}	Power	FlexRay/Ethernet 3.3 V I/O supply	164		D5	
V _{DD_HV_FLA}	Power	Decoupling supply pin for flash	155		D16, E15	
V _{DD_HV_ADV}	Power	High voltage supply for ADC	54		Y5	
V _{SS_HV_ADR_S}	Reference	Ground reference of ADC SAR	51		Y7	
V _{DD_HV_ADR_S}	Reference	Voltage reference of ADC SAR	52		Y6	
V _{SS_HV_ADR_D}	Reference	Ground reference of ADC Σ/Δ	27		T2	
V _{DD_HV_ADR_D}	Reference	Voltage reference of ADC Σ/Δ	26		T1	

4.4.3 System pins/balls

Table 8 contains information on system pin functions for the devices.

Table 8. System pins

Symbol	Description	Direction	QFP pin		BGA ball	
			176	FQ216	292PD	292ED
PORST	Power on reset with Schmitt trigger characteristics and noise filter. PORST is active low	Bidirectional	121		G17	
ESR0	External functional reset with Schmitt trigger characteristics and noise filter. ESR0 is active low	Bidirectional	122		F16	
TESTMODE	Pin for testing purpose only. An internal pull-down is implemented on the TESTMODE pin to prevent the device from entering TESTMODE. It is recommended to connect the TESTMODE pin to V _{SS_HV_IO} on the board. The value of the TESTMODE pin is latched at the negation of reset and has no affect afterward. The device will not exit reset with the TESTMODE pin asserted during power-up.	Input only	118		H19	
XTAL	Analog output of the oscillator amplifier circuit needs to be grounded if oscillator is used in bypass mode.	Output	103		M19	
EXTAL	Analog input of the oscillator amplifier circuit when oscillator is not in bypass mode Analog input for the clock generator when oscillator is in bypass mode	Input	102		M20	

4.4.4 LVDS pins/balls

The following table contains information on LVDS pin functions for the devices.

Table 9. LVDS pin descriptions

Functional block	Port pin	Signal	Signal description	Direction	QFP pin (176, FQ216)	BGA ball (292 PD, 292 ED)
SIPI / LFAST ⁽¹⁾	PF[13]	SIPI_RXN	Interprocessor Bus LFAST, LVDS Receive Negative Terminal	I	107	K19
	PD[7]	SIPI_RXP	Interprocessor Bus LFAST, LVDS Receive Positive Terminal	I	108	J19
	PD[6]	SIPI_TXN	Interprocessor Bus LFAST, LVDS Transmit Negative Terminal	O	109	K20
	PA[14]	SIPI_TXP	Interprocessor Bus LFAST, LVDS Transmit Positive Terminal	O	110	J20

Table 9. LVDS pin descriptions (continued)

Functional block	Port pin	Signal	Signal description	Direction	QFP pin (176, FQ216)	BGA ball (292 PD, 292 ED)
High-Speed Debug (HSD) / LFAST ^{(1),(2)}	PA[8]	DEBUG_TXN	Debug LFAST, LVDS Transmit Negative Terminal	O	111	H17
	PA[7]	DEBUG_TXP	Debug LFAST, LVDS Transmit Positive Terminal	O	112	K16
	PA[5]	DEBUG_RXN	Debug LFAST, LVDS Receive Negative Terminal	I	114	L19
	PA[9]	DEBUG_RXP	Debug LFAST, LVDS Receive Positive Terminal	I	113	H16
DSPI 4 Microsecond Bus	PD[3]	SCK	DSPI 4 Microsecond Bus Serial Clock, LVDS Negative Terminal	O	156	B12
	PD[2]	SCK	DSPI 4 Microsecond Bus Serial Clock, LVDS Positive Terminal	O	157	A12
	PD[1]	SOUT	DSPI 4 Microsecond Bus Serial Data, LVDS Negative Terminal	O	158	B11
	PD[0]	SOUT	DSPI 4 Microsecond Bus Serial Data, LVDS Positive Terminal	O	159	A11
DSPI 5 Microsecond Bus	PF[9]	SCK	DSPI 5 Microsecond Bus Serial Clock, LVDS Negative Terminal	O	95	P20
	PF[10]	SCK	DSPI 5 Microsecond Bus Serial Clock, LVDS Positive Terminal	O	96	P19
	PF[11]	SOUT	DSPI 5 Microsecond Bus Serial Data, LVDS Negative Terminal	O	97	R20
	PF[12]	SOUT	DSPI 5 Microsecond Bus Serial Data, LVDS Positive Terminal	O	98	R19

Table 9. LVDS pin descriptions (continued)

Functional block	Port pin	Signal	Signal description	Direction	QFP pin (176, FQ216)	BGA ball (292 PD, 292 ED)
Differential DSPI 2	PD[3]	SCK	Differential DSPI 2 Clock, LVDS Negative Terminal	O	156	B12
	PD[2]	SCK	Differential DSPI 2 Clock, LVDS Positive Terminal	O	157	A12
	PD[1]	SOUT	Differential DSPI 2 Serial Output, LVDS Negative Terminal	O	158	B11
	PD[0]	SOUT	Differential DSPI 2 Serial Output, LVDS Positive Terminal	O	159	A11
	PF[13]	SIN	Differential DSPI 2 Serial Input, LVDS Negative Terminal	I	107	K19
	PD[7]	SIN	Differential DSPI 2 Serial Input, LVDS Positive Terminal	I	108	J19
	PH[8]	SIN	Differential DSPI 2 Serial Input, LVDS Negative Terminal	I	151	D12
	PH[7]	SIN	Differential DSPI 2 Serial Input, LVDS Positive Terminal	I	152	D11

Table 9. LVDS pin descriptions (continued)

Functional block	Port pin	Signal	Signal description	Direction	QFP pin (176, FQ216)	BGA ball (292 PD, 292 ED)
Differential DSPI 5	PF[9]	SCK	Differential DSPI 5 Clock, LVDS Negative Terminal	O	95	P20
	PF[10]	SCK	Differential DSPI 5 Clock, LVDS Positive Terminal	O	96	P19
	PF[11]	SOUT	Differential DSPI 5 Serial Output, LVDS Negative Terminal	O	97	R20
	PF[12]	SOUT	Differential DSPI 5 Serial Output, LVDS Positive Terminal	O	98	R19
	PF[13]	SIN	Differential DSPI 5 Serial Input, LVDS Negative Terminal	I	107	K19
	PD[7]	SIN	Differential DSPI 5 Serial Input, LVDS Positive Terminal	I	108	J19
	PI[14]	SIN	Differential DSPI 5 Serial Input, LVDS Negative Terminal	I	105	K17
	PI[15]	SIN	Differential DSPI 5 Serial Input, LVDS Positive Terminal	I	106	J17

1. DRCLK and TCK/DRCLK usage for SIPI LFAST and Debug LFAST are described in the *SPC57EM80xx Microcontroller Reference Manual* SIPI LFAST and Debug LFAST chapters.
2. Pads use special enable signal from DCI block: DCI driven enable for Debug LFAST pads is transparent to user.

Table 10. Aurora pin descriptions

Functional Block	PAD	Signal	Signal Description	Direction	QFP		BGA	
					176	FQ216	292PD	292ED
Nexus Aurora High Speed Trace	—	TX0P	Nexus Aurora High Speed Trace Lane 0, LVDS Positive Terminal	O	—	A16	—	U14
	—	TX0N	Nexus Aurora High Speed Trace Lane 0, LVDS Negative Terminal	O	—	A15	—	U13
	—	TX1P	Nexus Aurora High Speed Trace Lane 1, LVDS Positive Terminal	O	—	A10	—	U12
	—	TX1N	Nexus Aurora High Speed Trace Lane 1, LVDS Negative Terminal	O	—	A9	—	U11
	—	TX2P	Nexus Aurora High Speed Trace Lane 2, LVDS Positive Terminal	O	—	A6	—	P11
	—	TX2N	Nexus Aurora High Speed Trace Lane 2, LVDS Negative Terminal	O	—	A5	—	P10
	—	TX3P	Nexus Aurora High Speed Trace Lane 3, LVDS Positive Terminal	O	—	A3	—	K7
	—	TX3N	Nexus Aurora High Speed Trace Lane 3, LVDS Negative Terminal	O	—	A2	—	L7
	—	CLKP (BD-AGBTCL KP)	Nexus Aurora High Speed Trace Clock, LVDS Positive Terminal	I	—	A19	—	U16
	—	CLKN (BD-AGBTCL KN)	Nexus Aurora High Speed Trace Clock, LVDS Negative Terminal	I	—	A18	—	U15

4.4.5 Generic pins/balls

The I/O Signal Description Table contains information on generic pins/balls, including the port pin, SIUL MSCR number, MSCR SSS, function, module, signal description, direction, pad type, and package pin/ball numbers for each I/O pin on the device.

Note: The SPC57EM80xx I/O Signal Description and Input Multiplexing Tables are contained in a Microsoft Excel® workbook file attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

Multi-function pins are programmable via their respective Multiplexed Signal Configuration Register (MSCR) Source Signal Select (SSS) values. The SSS value selects which source signal is connected to the associated destination.

For additional information, refer to [Chapter 16: System Integration Unit Lite2 \(SIUL2\)](#).

5 Memory Map

[Table 11](#) shows the device memory map for the SPC57EM80xx.

All addresses on the SPC57EM80xx, including those that are reserved, are identified in the table. The addresses represent the physical addresses assigned to each region or module name.

[Table 11](#) Summarizes the complete 32-bit address map. A memory access, either read or write, to any region marked as 'Reserved' will result in a Transfer Error. This error can generate an interrupt request in the CPU.

[Table 12](#) lists the memory map of the peripheral region. This region is sub-divided into 16 KB byte regions or 'slots'. A peripheral occupies 1 or more slots. A memory access, either read or write, to any slot marked as 'Reserved' will result in a Transfer Error. This error can generate an interrupt request in the CPU.

Within a 16 KB byte peripheral slot, it is normal for the peripheral registers to only occupy part of that 16 KB byte slot - normally from the lowest address upwards. The individual chapters for each block, define all the registers within that contiguous group of registers.

Any memory access outside the range of the defined register map will result in a transfer error. This error can generate an interrupt request in the CPU.

Tables [Table 13](#), [Table 14](#), and [Table 15](#) define the embedded memory maps and the buddy memory map. A memory access, either read or write, to any region marked as 'Reserved' in these tables, will result in a Transfer Error. This error can generate an interrupt request in the CPU.

Table 11. Overview Memory Map

Start Address	End Address	Allocated Size	Used Size	Description
Flash (XBAR port 0 - 0)				
0x00000000	0x003FFFFFFF	—		Reserved
0x00400000	0x00403FFFF	16 KB	16 KB	UTest NVM Block—no overlay
0x00404000	0x00407FFFF	16 KB	16 KB	BAF Flash low block - no overlay
0x00408000	0x005FFFFFFF	2016 KB	—	Reserved
0x00600000	0x0067FFFFF	512 KB	144 KB	HSM Code—no overlay
0x00680000	0x007FFFFFFF	1.5 MB	32 KB	HSM Data—no overlay
0x00800000	0x009FFFFFFF	22048 KB	256 KB	Data Flash Blocks—no overlay
0x00A00000	0x00FFFFFFF	6 MB	256 KB	Low and Mid Flash Blocks—no overlay
0x01000000	0x01FFFFFFF	16 MB	3584 KB	Large Flash Blocks—no overlay
0x02000000	0x089FFFFFFF	106 MB		Reserved
0x08A00000	0x08FFFFFFF	6 MB	256 KB	Low and mid flash blocks—overlay enabled mapping
0x09000000	0x09FFFFFFF	16 MB	3.5 MB	Large flash blocks—overlay enabled mapping
0x0A000000	0x0BFFFFFFF	32 MB		Reserved

Table 11. Overview Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	Description
0x0C000000	0x0C1FFFFFF	2 MB	1 MB	Extended Overlay RAM ⁽¹⁾
0x0C200000	0x0C7FFFFFF	6 MB		Reserved
0x0C800000	0x0CFFFFFF	8 MB	1 MB	Buddy Device Registers
0x0D000000	0x0D003FFF	16 KB	16 KB	Internal Overlay RAM
0x0D004000	0x0FFFFFFF	—		Reserved
XBAR Slave Port 0–1				
0x10000000	0x1FFFFFFF	256 MB		Reserved for EBI
XBAR Slave Port 0–2				
0x20000000	0x2FFFFFFF	256 MB		Reserved
Spare Port (XBAR Slave Port 0–3)				
0x30000000	0x3FFFFFFF	256 MB		Reserved
System RAM (XBAR Slave Port 0–4)				
0x40000000	0x401FFFFFF	2048 KB	128 KB	System RAM
0x40200000	0x4FFFFFFF	—		Reserved System RAM
Local Memory (XBAR Slave Port 0–5)⁽²⁾				
0x50000000	0x5FFFFFFF	256 MB		Processor Core local SRAM
Reserved Port A (XBAR Slave Port 0–6)				
0x60000000	0x6FFFFFFF	256 MB		Reserved
Reserved Port B (XBAR Slave Port 0–7)				
0x70000000	0x7FFFFFFF	256 MB		Reserved
Spare Port (XBAR Slave Port 1–8)				
0x80000000	0x8FFFFFFF	256 MB		Reserved
Spare Port (XBAR Slave Port 1–9)				
0x90000000	0x9FFFFFFF	256 MB		Reserved
HSM (Internal to HSM) (XBAR Slave Port 1–A)				
0xA0000000	0xA000FFFF	64 KB		Hardware Security Module (HSM)
0xA0010000	0xAFFFFFFF	262080 KB		Reserved
Spare Port (XBAR Slave Port 1–B)				
0xB0000000	0xBFFFFFFF	262144 KB		Reserved
Reserved Port C (XBAR Slave Port 1–C)				
0xC0000000	0xCFFFFFFF	262144 KB		Reserved

Table 11. Overview Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	Description
Reserved Port D (XBAR Slave Port 1–D)				
0xD0000000	0xDFFFFFFF	262144 KB		Reserved
Spare Port (XBAR Slave Port 1–E)				
0xE0000000	0xEFFFFFFF	262144 KB		Reserved
Peripherals PBRIDGE_A, PBRIDGE_B (XBAR Slave Port 1–F)				
0xF0000000	0xFFFFFFFF	256 MB		Peripheral address space ⁽³⁾

1. Address range is for extended overlay RAM for buddy device.

2. See [Table 15](#) for local SRAM addresses.

3. See [Table 12](#) for details.

Table 12. Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map

Start Address	End Address	Allocated Size	Used Size	PBRIDGE	PBRIDGE Access Control Register	Description
0xF0000000	0xF7FFFFFF	—				Reserved
0xF8000000	0xF8003FFF	16 KB		B	PACR0	Peripheral Bridge B (PBRIDGE_B)
0xF8004000	0xFBE03FFF	—				Reserved
0xFBE04000	0xFBE07FFF	16 KB		B	OPACR126	SARADC_1
0xFBE08000	0xFBE0BFFF	16 KB		B	OPACR125	SAR ADC 2 (SARADC_2)
0xFBE0C000	0xFBE0FFFF	16 KB		B	OPACR124	SAR ADC 3 (SARADC_3)
0xFBE10000	0xFBE17FFF	—				Reserved
0xFBE18000	0xFBE1BFFF	16 KB		B	OPACR121	SAR ADC 6 (SARADC_6)
0xFBE1C000	0xFBE1FFFF	16 KB		B	OPACR120	SAR ADC 7 (SARADC_7)
0xFBE20000	0xFBE3FFFF	—				Reserved
0xFBE40000	0xFBE43FFF	16 KB		B	OPACR111	Peripheral Sensor Interface 1 (PSI5_1)
0xFBE44000	0xFBE5BFFF	—				Reserved
0xFBE5C000	0xFBE5FFFF	16 KB		B	OPACR104	SENT (SAE J2716) Receiver 1 (SRX_1)
0xFBE60000	0xFBE6FFFF	—				Reserved
0xFBE70000	0xFBE73FFF	16 KB		B	OPACR99	Deserial Serial Peripheral Interface 2 (DSPI_2)
0xFBE74000	0xFBE77FFF	16 KB		B	OPACR98	Deserial Serial Peripheral Interface 3 (DSPI_3)

Table 12. Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFBE78000	0xFBE7BFFF	16 KB		B	OPACR97	Deserial Serial Peripheral Interface 5 (DSPI_5)
0xFBE7C000	0xFBE8BFFF	—				Reserved
0xFBE8C000	0xFBE8FFFF	16 KB		B	OPACR92	LIN Controller 2 (LINFlexD_2)
0xFBE90000	0xFBEA7FFF	—				Reserved
0xFBEA8000	0xFBEABFFF	16 KB		B	OPACR85	LIN Controller 15 (LINFlexD_15)
0xFBEAC000	0xFBF0BFFF	—				Reserved
0xFBF0C000	0xFBF0FFFF	16 KB		B	OPACR60	Sigma Delta (SD) ADC 1 (SDADC_1)
0xFBF10000	0xFBF13FFF	16 KB		B	OPACR59	Sigma Delta (SD) ADC 3 (SDADC_3)
0xFBF14000	0xFBF17FFF	16 KB		B	OPACR58	Sigma Delta (SD) ADC 5 (SDADC_5)
0xFBF18000	0xFBF57FFF	—				Reserved
0xFBF58000	0xFBF5BFFF	16 KB		B	OPACR41	Fault Collection and Control Unit (FCCU)
0xFBF5C000	0xFBF63FFF	—				Reserved
0xFBF64000	0xFBF67FFF	16 KB		B	OPACR38	Cyclic Redundancy Check 1 (CRC_1)
0xFBF68000	0xFBF73FFF	—				Reserved
0xFBF74000	0xFBF77FFF	16 KB		B	OPACR34	PSI5_S (PSI5 UART Module)
0xFBF78000	0xFBFB01FF	—				Reserved
0xFBFB0200	0xFBFB023F	64 Bytes		B	OPACR19	Clock Monitor Unit for Fast Crossbar (CMU_FXBAR)
0xFBFB0240	0xFBFB027F	64 Bytes		B	OPACR19	Clock Monitor Unit for Peripheral Shell Crossbar (CMU_SXBAR)
0xFBFB0280	0xFBFB02BF	64 Bytes		B	OPACR19	Clock Monitor Unit for Peripheral Bridge (CMU_PBRIDGE)
0xFBFB02C0	0xFBFB02FF	64 Bytes		B	OPACR19	Clock Monitor Unit for Peripherals (CMU_PER)
0xFBFB0300	0xFBFB03FF	64 Bytes		B	OPACR19	Clock Monitor Unit for Sigma Delta ADC (CMU_ADCSD)
0xFBFB0340	0xFBFB037F	64 Bytes		B	OPACR19	Clock Monitor Unit for SAR ADC (CMU_ADCSAR)
0xFBFB0380	0xFBFB03BF	64 Bytes		B	OPACR19	Clock Monitor Unit for SENT (CMU_SENT)
0xFBFB03C0	0xFBFB03FF	64 Bytes		B	OPACR19	Clock Monitor Unit for PSI5 (CMU_PSI5_F189)

Table 12. Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFBFB0400	0xFBFB043F	64 Bytes		B	OPACR19	Clock Monitor Unit for PSI5 (CMU_PSI5_F125)
0xFBFB0440	0xFBFB047F	64 Bytes		B	OPACR19	Clock Monitor Unit for PSI5 (CMU_PSI5_1μS)
0xFBFB0480	0xFBFFFFFF	—	Reserved			
0xFC000000	0xFC003FFF	16 KB		A	PACR0	Peripheral Bridge A (PBRIDGE_A)
0xFC004000	0xFC007FFF	16 KB		A	PACR1	Computational Shell Crossbar (XBAR_0)
0xFC008000	0xFC00BFFF	16 KB		A	PACR2	Peripheral Shell Crossbar (XBAR_1)
0xFC00C000	0xFC00FFFF	—	Reserved			
0xFC010000	0xFC013FFF	16 KB		A	PACR4	System Memory Protection Unit 0 (SMPU_0)
0xFC014000	0xFC017FFF	16 KB		A	PACR5	System Memory Protection Unit 1 (SMPU_1)
0xFC018000	0xFC01BFFF	16 KB		A	PACR6	Crossbar Integrity Checker 0 (XBIC0)
0xFC01C000	0xFC01FFFF	16 KB		A	PACR7	Crossbar Integrity Checker 1 (XBIC1)
0xFC020000	0xFC023FFF	16 KB		A	PACR8	PRAM (Platform RAM controller)
0xFC024000	0xFC027FFF	—	Reserved			
0xFC028000	0xFC02BFFF	16 KB		A	PACR10	Platform Configuration Module (PCM)
0xFC02C000	0xFC02FFFF	—	Reserved			
0xFC030000	0xFC033FFF	16 KB		A	PACR12	PFLASH (Platform Flash controller)
0xFC034000	0xFC03BFFF	—	Reserved			
0xFC03C000	0xFC03FFFF	16 KB		A	PACR15	Semaphore (SEMA4)
0xFC040000	0xFC043FFF	16 KB		A	PACR16	Interrupt Controller (INTC0)
0xFC044000	0xFC04FFFF	—	Reserved			
0xFC050000	0xFC053FFF	16 KB		A	PACR20	Software Watchdog Timer 0 (SWT_0)
0xFC054000	0xFC057FFF	16 KB		A	PACR21	Software Watchdog Timer 1 (SWT_1)
0xFC058000	0xFC05BFFF	16 KB		A	PACR22	Software Watchdog Timer 2 (SWT_2)

Table 12. Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFC05C000	0xFC05FFFF	16 KB		A	PACR23	Software Watchdog Timer 3 (SWT_3)
0xFC060000	0xFC067FFF	—	Reserved			
0xFC068000	0xFC06BFFF	16 KB		A	PACR26	System Timer Module 0 (STM_0)
0xFC06C000	0xFC06FFFF	16 KB		A	PACR27	System Timer Module 1 (STM_1)
0xFC070000	0xFC073FFF	16 KB		A	PACR28	System Timer Module 2 (STM_2)
0xFC074000	0xFC09FFFF	—	Reserved			
0xFC0A0000	0xFC0A3FFF	16 KB		A	PACR40	Direct Memory Access Controller 0 (DMA_0)
0xFC0A4000	0xFC0AFFFF	—	Reserved			
0xFC0B0000	0xFC0B3FFF	16 KB		A	PACR44	Fast Ethernet Controller (FEC_0)
0xFC0B4000	0xFC0BFFFF	—	Reserved			
0xFC0C0000	0xFC0C3FFF	16 KB	AMU	A	PACR48	ASC Modeling Unit (AMU) Instruction Memory
0xFC0C4000	0xFC0C7FFF	16 KB		A	PACR49	ASC Modeling Unit (AMU) Instruction Memory
0xFC0C8000	0xFC0CBFFF	16 KB		A	PACR50	ASC Modeling Unit (AMU) Instruction Memory
0xFC0CC000	0xFC0CFFFF	16 KB		A	PACR51	ASC Modeling Unit (AMU) Instruction Memory
0xFC0D0000	0xFC0D3FFF	16 KB		A	PACR52	ASC Modeling Unit (AMU) Registers
0xFC0D4000	0xFFCFFFFF	—	Reserved			
0xFFD00000	0xFFDBFFFF	768 KB		A	OPACR144 – OPACR191	Generic Timer Unit 103 Integration Module (GTMINT)
0xFFDC0000	0xFFDFFFFF	—	Reserved			
0xFFE00000	0xFFE03FFF	16 KB		A	OPACR127	SARADC_0
0xFFE04000	0xFFE0FFFF	—	Reserved			
0xFFE10000	0xFFE13FFF	16 KB		A	OPACR123	SAR ADC 4 (SARADC_4)
0xFFE3C000	0xFFE3FFFF	16 KB		A	OPACR112	SARADC_B
0xFFE40000	0xFFE43FFF	16 KB		A	OPACR111	Peripheral Sensor Interface 0 (PSI5_0)
0xFFE44000	0xFFE4FFFF	—	Reserved			
0xFFE50000	0xFFE53FFF	16 KB		A	OPACR107	FlexRay Communication Controller 0 (FLEXRAY_0)
0xFFE54000	0xFFE5BFFF	—	Reserved			

Table 12. Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFFE5C000	0xFFE5FFFF	16 KB		A	OPACR104	SENT (SAE J2716) Receiver 0 (SRX_0)
0xFFE60000	0xFFE67FFF	—				Reserved
0xFFE68000	0xFFE6BFFF	16 KB		A	OPACR101	Inter IC Bus 0 (IIC_0)
0xFFE6C000	0xFFE6FFFF	—				Reserved
0xFFE70000	0xFFE73FFF	16 KB		A	OPACR99	Deserial Serial Peripheral Interface 0 (DSPI_0)
0xFFE74000	0xFFE77FFF	16 KB		A	OPACR98	Deserial Serial Peripheral Interface 1 (DSPI_1)
0xFFE78000	0xFFE7BFFF	16 KB		A	OPACR97	Deserial Serial Peripheral Interface 4 (DSPI_4)
0xFFE7C000	0xFFE87FFF	—				Reserved
0xFFE88000	0xFFE8BFFF	16 KB		A	OPACR93	Deserial Serial Peripheral Interface 12 (DSPI_12)
0xFFE8C000	0xFFE8FFFF	16 KB		A	OPACR92	LIN Controller 0 (LINFlexD_0)
0xFFE90000	0xFFE93FFF	16 KB		A	OPACR91	LIN Controller 1 (LINFlexD_1)
0xFFE94000	0xFFEA7FFF	—				Reserved
0xFFEA8000	0xFFEABFFF	16 KB		A	OPACR85	LIN Controller 14 (LINFlexD_14)
0xFFEAC000	0xFFED3FFF	—				Reserved
0xFFED4000	0xFFED7FFF	16 KB	15.5 KB	A	OPACR74	Shared CAN Message RAM
0xFFED8000	0xFFEDBFFF	—				Reserved
0xFFEDC000	0xFFEDFFFF	16 KB		A	OPACR72	Time Triggered Controller Area Network (M_TTCAN_0)
0xFFEE0000	0xFFEE3FFF	—				Reserved
0xFFEE4000	0xFFEE7FFF	16 KB		A	OPACR70	CAN Sub-System: Controller Area Network 1 (M_CAN_1)
0xFFEE8000	0xFFEEBFFF	16 KB		A	OPACR69	CAN Sub-System: Controller Area Network 2 (M_CAN_2)
0xFFEEC000	0xFFEEFFFF	16 KB		A	OPACR68	CAN Sub-System: Controller Area Network 3 (M_CAN_3)
0xFFEF0000	0xFFFF0BFFF	—				Reserved
0xFFFF0C000	0xFFFF0FFFF	16 KB		A	OPACR60	Sigma Delta (SD) ADC 0 (SDADC_0)
0xFFFF10000	0xFFFF13FFF	16 KB		A	OPACR59	Sigma Delta (SD) ADC 2 (SDADC_2)

Table 12. Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFFFF14000	0xFFFF17FFF	16 KB		A	OPACR58	Sigma Delta (SD) ADC 4 (SDADC_4)
0xFFFF18000	0xFFFF2FFFF	—	Reserved			
0xFFFF30000	0xFFFF33FFF	16 KB		A	OPACR51	Hardware Security Module (HSM) Interface ⁽¹⁾
0xFFFF34000	0xFFFF37FFF	16 KB		A	OPACR50	Tamper Detection Module (TDM)
0xFFFF38000	0xFFFF3BFFF	16 KB		A	OPACR49	Development Trigger Semaphore (DTS)
0xFFFF3C000	0xFFFF3FFFF	16 KB		A	OPACR48	JTAG Data Communication (JDC) Module
0xFFFF40000	0xFFFF43FFF	—	Reserved			
0xFFFF44000	0xFFFF47FFF	16 KB		A	OPACR46	Self Test Control Unit (STCU2)
0xFFFF48000	0xFFFF4BFFF	16 KB		A	OPACR45	JTAG Master (JTAGM)
0xFFFF4C000	0xFFFF4FFFF	—	Reserved			
0xFFFF50000	0xFFFF53FFF	16 KB		A	OPACR43	Memory Error Management Unit (MEMU)
0xFFFF54000	0xFFFF57FFF	16 KB		A	OPACR42	Indirect Memory Access Module (IMA)
0xFFFF58000	0xFFFF63FFF	—	Reserved			
0xFFFF64000	0xFFFF67FFF	16 KB		A	OPACR38	Cyclic Redundancy Check 0 (CRC_0)
0xFFFF68000	0xFFFF6BFFF	—	Reserved			
0xFFFF6C000	0xFFFF6C1FF	512 Bytes		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer (DMAMUX_0)
0xFFFF6C200	0xFFFF6C3FF	512 Bytes		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer (DMAMUX_1)
0xFFFF6C400	0xFFFF6C5FF	512 Bytes		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer (DMAMUX_2)
0xFFFF6C600	0xFFFF6C7FF	512 Bytes		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer (DMAMUX_3)
0xFFFF6C800	0xFFFF6C9FF	512 Bytes		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer (DMAMUX_4)

Table 12. Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFFFF6CA00	0xFFFF6CBFF	512 Bytes		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer (DMAMUX_5)
0xFFFF6CC00	0xFFFF7FFFF	—				Reserved
0xFFFF80000	0xFFFF83FFF	16 KB		A	OPACR31	Periodic Interval Timer 1 (PIT_1)
0xFFFF84000	0xFFFF87FFF	16 KB		A	OPACR30	Periodic Interval Timer 0 (PIT_0)
0xFFFF88000	0xFFFF97FFF	—				Reserved
0xFFFF98000	0xFFFF9BFFF	16 KB		A	OPACR25	Wake-Up Unit (WKPU)
0xFFFF9C000	0xFFFF9FFFF	—				Reserved
0xFFFFA0000	0xFFFFA03FF	1 KB		A	OPACR23	Power Control Unit (MC_PCU)
0xFFFFA0400	0xFFFFA3FFF	15 KB		A	OPACR23	Power Management Controller (PMCDIG)
0xFFFFA4000	0xFFFFA7FFF	—				Reserved
0xFFFFA8000	0xFFFFABFFF	16 KB		A	OPACR21	Reset Generation Module (MC_RGM)
0xFFFFAC000	0xFFFFAFFFF	—				Reserved
0xFFFFB0000	0xFFFFB003F	64 Byte		A	OPACR19	RCOSC (Internal RC Oscillator)
0xFFFFB0040	0xFFFFB007F	—				Reserved
0xFFFFB0080	0xFFFFB00BF	64 Byte		A	OPACR19	XOSC (Externally driven oscillator)
0xFFFFB00C0	0xFFFFB00FF	—				Reserved
0xFFFFB0100	0xFFFFB013F	64 Byte		A	OPACR19	FM Dual PLL (PLL)
0xFFFFB0140	0xFFFFB01FF	—				Reserved
0xFFFFB0200	0xFFFFB023F	64 Byte		A	OPACR19	Clock Monitor Unit for the PLL (CMU_PLL)
0xFFFFB0240	0xFFFFB06FF	—				Reserved
0xFFFFB0700	0xFFFFB3FFF	14 KB		A	OPACR19	Clock Generation Module (MC_CGM)
0xFFFFB4000	0xFFFFB7FFF	—				Reserved
0xFFFFB8000	0xFFFFBBFFF	16 KB		A	OPACR17	Mode Entry Module (MC_ME)
0xFFFFBC000	0xFFFFBFFFF	—				Reserved
0xFFFFC0000	0xFFFFC3FFF	16 KB		A	OPACR15	System Integration Unit Lite 2 (SIUL2)
0xFFFFC4000	0xFFFFCFFFF	—				Reserved
0xFFFFD0000	0xFFFFD3FFF	16 KB		A	OPACR11	Serial Interprocessor Interface 0 (SIPI_0)

Table 12. Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map (continued)

Start Address	End Address	Allocated Size	Used Size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFFFFD4000	0xFFFFD7FFF	16 KB		A	OPACR10	Serial Interprocessor Interface 1 (SIPI_1) (Debug)
0xFFFFD8000	0xFFFFDBFFF	16 KB		A	OPACR9	LFAST 0 (LFAST_0) (interprocessor communication)
0xFFFFDC000	0xFFFFDFFFF	16 KB		A	OPACR8	LFAST 1 (LFAST_1) (Debug)
0xFFFE0000	0xFFFE3FFF	16 KB		A	OPACR7	Flash main control registers
0xFFFE4000	0xFFFE7FFF	16 KB		A	OPACR6	Flash Alternate Control Registers
0xFFFE8000	0xFFFF3FFF	—	Reserved			
0xFFFF4000	0xFFFF7FFF	16 KB		A	OPACR2	Password and Device Security Module (PASS)
0xFFFF8000	0xFFFFBFFF	16 KB		A	OPACR1	System Status and Configuration Module (SSCM)
0xFFFFC000	0xFFFFFFF	16 KB		A	OPACR0	Boot Assist ROM (BAR)

1. This is the address space for the Host/HSM interface registers. See the SPC57EM80xx Microcontroller Security Reference Manual for details.

Table 13 is the detailed flash memory map for the SPC57EM80xx.

Table 13. Flash memory and overlay RAM map

Start Address	End Address	Allocated Size [KB]	Complete Flash Block Structure	Address Space	RWW Partition	Block size
Reserved—no overlay						
0x00000000	0x003FFFFFF	—	Reserved			
UTest NVM Block—no overlay						
0x00400000	0x00403FFF	16 KB	UTest NVM Block Space 16 KB	Low	0	16 KB
Low Flash Block—no overlay						
0x00404000	0x00407FFF	16 KB	BAF—16 KB flash block0	Low	0	16 KB
0x00408000	0x005FFFFFF	—	Reserved			
HSM Code—no overlay						
0x00600000	0x0060BFFF	48 KB	Reserved			
0x0060C000	0x0060FFFF	16 KB	16 KB HSM code block5	Low	1	16 KB
0x00610000	0x0061FFFF	64 KB	64 KB HSM code block2	Low	1	64 KB
0x00620000	0x0062FFFF	64 KB	64 KB HSM code block3	Low	1	64 KB
0x00630000	0x0067FFFF	—	Reserved			

Table 13. Flash memory and overlay RAM map (continued)

Start Address	End Address	Allocated Size [KB]	Complete Flash Block Structure	Address Space	RWW Partition	Block size
HSM Data—no overlay						
0x00680000	0x00683FFF	16 KB	16 KB HSM data EEPROM block0	Mid	2	16 KB
0x00684000	0x00687FFF	16 KB	16 KB HSM data EEPROM block1	Mid	3	16 KB
0x00688000	0x007FFFFFFF	—	Reserved			
Data Flash—no overlay						
0x00800000	0x0080FFFF	64 KB	EEPROM block0	High	4	64 KB
0x00810000	0x0081FFFF	64 KB	EEPROM block1	High	4	64 KB
0x00820000	0x0082FFFF	64 KB	EEPROM block2	High	5	64 KB
0x00830000	0x0083FFFF	64 KB	EEPROM block3	High	5	64 KB
0x00840000	0x009FFFFFFF	—	Reserved			
Low & Mid Flash Blocks—no overlay						
0x00A00000	0x00FBFFFF	—	Reserved			
0x00FC0000	0x00FC3FFF	16 KB	16 KB Flash block1	Low	0	16 KB
0x00FC4000	0x00FC7FFF	16 KB	16 KB Flash block2	Low	0	16 KB
0x00FC8000	0x00FCBFFF	16 KB	16 KB Flash block3	Low	1	16 KB
0x00FCC000	0x00FCFFFF	16 KB	16 KB Flash block4	Low	1	16 KB
0x00FD0000	0x00FD7FFF	32 KB	32 KB Flash block0	Low	0	32 KB
0x00FD8000	0x00FDFFFF	32 KB	32 KB Flash block1	Low	1	32 KB
0x00FE0000	0x00FEFFFF	64 KB	64 KB Low Flash block0	Low	0	64 KB
0x00FF0000	0x00FFFFFF	64 KB	64 KB Low Flash block1	Low	0	64 KB
Large Flash Blocks—no overlay						
0x01000000	0x0103FFFF	256 KB	256 KB Flash block0	256 KB	6	256 KB
0x01040000	0x0107FFFF	256 KB	256 KB Flash block1	256 KB	6	256 KB
0x01080000	0x010BFFFF	256 KB	256 KB Flash block2	256 KB	6	256 KB
0x010C0000	0x010FFFFF	256 KB	256 KB Flash block3	256 KB	6	256 KB
0x01100000	0x0113FFFF	256 KB	256 KB Flash block4	256 KB	6	256 KB
0x01140000	0x0117FFFF	256 KB	256 KB Flash block5	256 KB	6	256 KB
0x01180000	0x011BFFFF	256 KB	256 KB Flash block6	256 KB	6	256 KB
0x011C0000	0x011FFFFF	256 KB	256 KB Flash block7	256 KB	7	256 KB
0x01200000	0x0123FFFF	256 KB	256 KB Flash block8	256 KB	7	256 KB
0x01240000	0x0127FFFF	256 KB	256 KB Flash block9	256 KB	7	256 KB
0x01280000	0x012BFFFF	256 KB	256 KB Flash block10	256 KB	7	256 KB

Table 13. Flash memory and overlay RAM map (continued)

Start Address	End Address	Allocated Size [KB]	Complete Flash Block Structure	Address Space	RWW Partition	Block size
0x012C0000	0x012FFFFFFF	256 KB	256 KB Flash block11	256 KB	7	256 KB
0x01300000	0x0133FFFFFF	256 KB	256 KB Flash block12	256 KB	7	256 KB
0x01340000	0x0137FFFFFF	256 KB	256 KB Flash block13	256 KB	7	256 KB
0x01380000	0x01FFFFFFF	—	Reserved			
Reserved Flash—no overlay						
0x02000000	0x07FFFFFFF	—	Reserved			
Reserved Flash—overlay enabled mapping						
0x08000000	0x083FFFFFFF	—	Reserved			
Reserved Flash—overlay enabled mapping						
0x08400000	0x089FFFFFFF	—	Reserved			
Low & Mid flash blocks—overlay enabled mapping (Mirror of flash blocks starting at 0x00FC_0000 but with RAM overlay allowed)						
0x08A00000	0x08FBFFFFF	—	Reserved			
0x08FC0000	0x08FC3FFF	16 KB	16 KB Flash block1	Low	0	16 KB
0x08FC4000	0x08FC7FFF	16 KB	16 KB Flash block2	Low	0	16 KB
0x08FC8000	0x08FCBFFF	16 KB	16 KB Flash block3	Low	1	16 KB
0x08FCC000	0x08FCFFFF	16 KB	16 KB Flash block4	Low	1	16 KB
0x08FD0000	0x08FD7FFF	32 KB	32 KB Flash block0	Low	0	32 KB
0x08FD8000	0x08FDFFFF	32 KB	32 KB Flash block1	Low	1	32 KB
0x08FE0000	0x08FEFFFF	64 KB	64 KB Low Flash block0	Low	0	64 KB
0x08FF0000	0x08FFFFFFF	64 KB	64 KB Low Flash block1	Low	0	64 KB
Large flash blocks—overlay enabled mapping						
0x09000000	0x0903FFFF	256 KB	256 KB Flash block0	256 KB	6	256 KB
0x09040000	0x0907FFFF	256 KB	256 KB Flash block1	256 KB	6	256 KB
0x09080000	0x090BFFFF	256 KB	256 KB Flash block2	256 KB	6	256 KB
0x090C0000	0x090FFFFF	256 KB	256 KB Flash block3	256 KB	6	256 KB
0x09100000	0x0913FFFF	256 KB	256 KB Flash block4	256 KB	6	256 KB
0x09140000	0x0917FFFF	256 KB	256 KB Flash block5	256 KB	6	256 KB
0x09180000	0x091BFFFF	256 KB	256 KB Flash block6	256 KB	6	256 KB
0x091C0000	0x091FFFFF	256 KB	256 KB Flash block7	256 KB	7	256 KB
0x09200000	0x0923FFFF	256 KB	256 KB Flash block8	256 KB	7	256 KB
0x09240000	0x0927FFFF	256 KB	256 KB Flash block9	256 KB	7	256 KB
0x09280000	0x092BFFFF	256 KB	256 KB Flash block10	256 KB	7	256 KB

Table 13. Flash memory and overlay RAM map (continued)

Start Address	End Address	Allocated Size [KB]	Complete Flash Block Structure	Address Space	RWW Partition	Block size
0x092C0000	0x092FFFFFFF	256 KB	256 KB Flash block11	256 KB	7	256 KB
0x09300000	0x0933FFFFFF	256 KB	256 KB Flash block12	256 KB	7	256 KB
0x09340000	0x0937FFFFFF	256 KB	256 KB Flash block13	256 KB	7	256 KB
0x09380000	0x09FFFFFFF	—	Reserved			
Reserved flash—overlay enabled mapping						
0x0A000000	0x0A7FFFFFFF	—	Reserved			
Buddy Overlay RAM						
0x0C000000	0x0C1FFFFFFF	2 MB	Extended Overlay RAM		—	1 MB ⁽¹⁾
0x0C200000	0x0C7FFFFFFF	6 MB	Reserved			
Internal Overlay RAM						
0x0D000000	0x0D003FFF	16 KB	Internal Overlay RAM			16 KB
0x0D004000	0x0FFFFFFF	—	Reserved			

1. Consists of four 256 KB regions.

Table 14 is the memory map for the SPC57EM80xx buddy device.

Table 14. Buddy device memory map

Start Address	End Address	Allocated Size [KB]	Description	BD1M	BD2M
Buddy Overlay RAM					
0x0C000000	0x0C0FFFFFFF	1024 KB	Extended Overlay RAM	1M	1M
0x0C100000	0x0C1FFFFFFF	1024 KB	Reserved Buddy Overlay RAM		1M
0x0C200000	0x0C7FFFFFFF	—	Reserved		
Buddy Registers					
0x0C800000	0x0C803FFF	16 KB	System Integration Unit Lite 2 (SIUL2)		
0x0C804000	0x0C807FFF	16 KB	LFAST		
0x0C808000	0x0C80BFFF	16 KB	JTAGM		
0x0C80C000	0x0C80FFFF	16 KB	Reserved		
0x0C810000	0x0C813FFF	16 KB	Buddy Device Crossbar		
0x0C814000	0x0CFFFFFFF	—	Reserved		

The following table details the system RAM memory map.

Table 15. RAM memory map

Start Address	End Address	Allocated Size [KB]	Used	Description
System RAM (XBAR Port 0–4)				
0x40000000	0x4000FFFF	64 KB	64 KB	Standby RAM
0x40010000	0x4001FFFF	64 KB	64 KB	System RAM
0x40200000	0x4FFFFFFF	—	—	Reserved
Local Memory (XBAR Port 0–5)				
0x50000000	0x5000FFFF	64 KB	16 KB	I-MEM CPU0
0x50010000	0x507FFFFF	—	—	Reserved
0x50800000	0x5080FFFF	64 KB	64 KB	D-MEM CPU0
0x50810000	0x50FFFFFF	—	—	Reserved
0x51000000	0x5100FFFF	64 KB	16 KB	I-MEM CPU1
0x51010000	0x517FFFFF	—	—	Reserved
0x51800000	0x5180FFFF	64 KB	64 KB	D-MEM CPU1
0x51810000	0x51FFFFFF	8128 KB	—	Reserved
0x52000000	0x5200FFFF	64 KB	16 KB	I-MEM CPU2
0x52010000	0x527FFFFF	8128 KB	—	Reserved
0x52800000	0x5280FFFF	64 KB	64 KB	D-MEM CPU2
0x52810000	0x5FFFFFFF	8128 KB	—	Reserved

Table 16. UTest flash memory map

Start Address	End Address	Allocated Size [bytes]	Description	Comments
UTest Flash				
0x00400000	0x00400001	2	Temperature Sensor Calibration A	Contains the fitting parameter TSCA extracted during factory-calibration to be used by software on temperature formula in order to calculate die temperature (see formula in the Temperature Sensor Chapter).
0x00400002	0x00400003	2	Temperature Sensor Calibration B	Contains the fitting parameter TSCB extracted during factory-calibration to be used by software on temperature formula in order to calculate die temperature (see formula Temperature Sensor Chapter).

Table 16. UTest flash memory map (continued)

Start Address	End Address	Allocated Size [bytes]	Description	Comments
0x00400004	0x0040000B	8	Reserved	
0x0040000C	0x0040000F	4	Test Mode Disable Seal	Programmed by the customer to disable factory test mode access
0x00400010	0x0040001F	16	Test Mode Disable Block Select Group A	Programmed by the customer to select which blocks can NOT be tested in Failure Analysis
0x00400020	0x0040002F	16	Factory Erase diary Location	Programmed by the customer before one-time factory erase is allowed
0x00400030	0x0040003F	16	Test Mode Disable Block Select Group B	Secondary location that can be programmed by the customer to select which blocks can NOT be tested in Failure Analysis
0x00400040	0x0040005F	32	Customer Single Bit Correction Area	Programmed by factory to include ECC/EDC errors to allow testing of ECC/EDC hardware
0x00400060	0x0040007F	32	Customer Double Bit Detection Area	256 bits — Reserved for Double Bit Detection (Double bit data programmed by STMicroelectronics).
0x00400080	0x0040009F	32	Flash ECC Error Words Area	256 bits — Reserved for EDC after ECC Detection (Data programmed by STMicroelectronics).
0x004000A0	0x004000BF	32	Unique Identifier (UID)	Programmed during factory test. The UID includes information on Wafer lot, X/Y-position on the wafer, manufacturing data, test results, company ID: 66 = STMicroelectronics
0x004000C0	0x004000C3	4	Soft DCF Record Start Address	Programmed by the customer to indicate a non-standard start address for DCF Records targeting 'Soft' DCF clients
0x004000C4	0x004000FF	60	Reserved	

Table 16. UTest flash memory map (continued)

Start Address	End Address	Allocated Size [bytes]	Description	Comments
0x00400100	0x00400103	4	Test Mode Override Passcode	Protected from read operations by flash BIU from Production at OEM Life Cycle. Programmed by customer. Becomes unreadable (hidden) as soon as customer programs lifecycle stage to 'Production at OEM' or later
0x00400104	0x0040011F	28	Reserved	
0x00400120	0x0040013F	32	JTAG Password	
0x00400140	0x0040015F	32	PASS Password Group 0	
0x00400160	0x0040017F	32	PASS Password Group 1	
0x00400180	0x0040019F	32	PASS Password Group 2	
0x004001A0	0x004001BF	32	PASS Password Group 3	
0x004001C0	0x004001FF	64	Reserved	
0x00400200	0x0040020F	16	Lifecycle Slot 0 - JDP_PROD	"JDP Production" Lifecycle slot. Programmed by JDP during test
0x00400210	0x0040021F	16	Lifecycle Slot 1 - CUST_DEL	"Customer Delivery" Lifecycle slot. Programmed by JDP during test
0x00400220	0x0040022F	16	Lifecycle Slot 2 - OEM_PROD	"OEM Production" Lifecycle slot. Programmed by customer during ECO module production/test
0x00400230	0x0040023F	16	Lifecycle Slot 3 - IN_FIELD	"In Field" Lifecycle slot. Programmed by customer during ECO module production/test
0x00400240	0x0040024F	16	Lifecycle Slot 4 - FA	Failure Analysis Lifecycle slot. Programmed by the customer on returned modules
0x00400250	0x004002FF	176	Reserved	
0x00400300	0x00400307	8	DCF Start Record	DCF start Record. Programmed by JDP during Test. Contiguous list of DCF records starts at this address
0x00400308	0x00400347	64	DCF HSM 'ROM' key Records	Space for 8 DCF records used to load the HSM 'ROM' keys. <i>Note: This region is only fully readable while Lifecycle = JDP_PROD. When Lifecycle has reached CUST_DEL, this region is read protected by the flash memory flash controller (PFLASH). Reads to this region return dummy DCF records, allowing software to parse the complete DCF record list, from start record to stop record</i>

Table 16. UTest flash memory map (continued)

Start Address	End Address	Allocated Size [bytes]	Description	Comments
0x00400348	0x00400FFF	3584	DCF Records	Contiguous list of DCF Records beginning at 0x00400200. Initial records are programmed at the factory. Subsequent records are added to the end of the list by the customer
0x00401000	0x00403FFF	12288	Reserved for customer OTP data	Programmed by the customer
0x00404000	0x00407FFF	16384	BAF code	Reserved for Boot Assist Flash (BAF). This is factory code that executes every reset to implement boot features. It is programmed at the factory and is OTP

6 Functional Safety

6.1 Introduction

The SPC57EM80xx offers a set of features to support applications that need to fulfill functional safety requirements as defined by automotive safety integrity level ASIL D of ISO 26262. This chapter gives an overview of the safety implementation for the SPC57EM80xx, and discusses the operation of the safety mechanisms.

6.2 Safety overview

The SPC57EM80xx has been designed for automotive applications executing safety-relevant tasks that require a fail-silent or fail-indicate MCU. The SPC57EM80xx supports a Fault Tolerant Time Interval (FTTI) of at least 10 ms. Shorter intervals are possible depending on the actual combination of failure detection and signaling mechanisms as documented in the Safety Manual (SM), but the SPC57EM80xx generally supports an FTTI of 10 ms.

6.3 Module categorization

The definition of the SPC57EM80xx safety concept and its validation (particularly the computation of Probabilistic Metric for Random Hardware Failures (PMHF)) has been done by splitting system modules into four categories as shown in [Table 17](#).

Table 17. PMHF module categories

Module category	Category abbreviation	Description
Vital system modules	ViMo	All modules that can directly influence the correct operation of the software execution (by influence of the correct function of CPU, RAM, flash, and the bus interconnect). Examples of ViMo are the core, the cache, the crossbar, and the interrupt controller (INTC), as well as clock or power generation.
Peripheral modules	PeMo	All microcontroller modules involved in I/O operation, specifically including the I/O bridge.
Non-safety modules	NoSaMo	Modules that belong in one of the above categories but that do not require hardware-level safety measures. Examples include the DMA, the performance core, and the I/O core. Only their non-interference with modules from the other categories is ensured by hardware. In case they are used in a safety-relevant way, application measures are necessary to ensure their safe operation.
Other support modules	SuMo	These are modules that, although safety-relevant and in principle potential members of the ViMo category, cannot lead to a violation of the safety goal on their own. Typically, these will only have multiple point faults, but not single point faults. Example SuMo's are the RCCU and Clock Monitoring Unit (CMU).

[Table 18](#) shows module classifications for the SPC57EM80xx.

Table 18. Module classification

Classification	Modules
ViMos	Master Safety Core, Master Safety Core D-Cache controller, Master Safety Core I-Cache controller, Local Memory Controllers Master Core, Safety Core D-Cache array, Safety Core I-Cache array, Safety Core Local D-Mem array, Safety Core Local I-Mem array, INTC, XBAR_0 (Computational), SRAM controller, system RAM array, flash memory controller, flash memory array, overlay RAM, system timer, MC_RGM, Dual PLL Digital, XOSC, MC_CGM, MC_ME, voltage regulator, MC_PMC, STCU2, SSCM, PIT
PeMos	SAR ADC, PSI5, SENT, LINFlexD, DSPI, M_CAN/TT_CAN, FlexRay, FEC, SIUL2, XBAR_1 (Peripheral), Concentrator FEC, FlexRay, and LFAST, Sigma-Delta ADC PBRIDGE n

Table 18. Module classification (continued)

Classification	Modules
SuMos	RCCU, Checker Safety Core, System RAM/local memory controllers of Checker Safety Core (overall 4), Checker Safety Core D-Cache controller, Checker Safety Core I-Cache controller, CRC units, Temperature Sensors, IRCOSC, CMUs, HVDs, LVDs, Bandgap reference, LBIST controllers, IMA, REG_PROT, SWT, FCCU, MEMU
NoSaMos	[SCG18.016][Covers: ASaf1061]Performance Core (Core_1)[end], Peripheral Core (Core_2), D-Cache Performance Core (Core_1), I- Cache Performance Core (Core_1), I-Cache Peripheral Core (Core_2), other local memories of Performance Core (Core_1) and Peripheral Core (Core_2), EBI (LFAST, SIPI), eDMA, DMACHMUX, GTM, Nexus, JTAG, Concentrator HSM and DMA, I2C, SEMA4, WKPU SPU

The SPC57EM80xx implements no specific protection of NoSaMos against Common Cause Failures between them.

6.4 System implementation

6.4.1 General concept

In general, safety integrity is achieved in the following ways:

- For the Safety Core and its closely related periphery, safety is ensured by a delayed lockstep approach (see [Section 6.4.2, Dual-core lockstep](#)).
- The safety of storage and of the data path to storage and periphery is ensured by End-to-End ECC (E2E ECC) with address encoding and selected additional measures for individual modules. For the periphery, end-to-end ends at the I/O bridges (see [Section 6.4.1.1, End-to-End protection on data path](#)).
- Clock and power, generation and distribution, are supervised by dedicated monitors (see [Section 6.4.1.2, Clock and power monitoring](#)).
- The safety of the periphery is ensured by application-level measures (such as connecting one sensor to different I/O modules, sensor validation by sensor fusion, and so on). Hardware supports this application-level redundancy by providing redundant I/O modules connected to different peripheral bridges (PBRIDGE) to maximize the independence between the monitored and monitoring resources (see [Section 6.4.1.3, I/O peripherals](#), and [Section 6.4.1.4, Communication controllers](#)).
- MBIST and LBIST are provided to avoid the accumulation of latent faults in the functional logic as well as the safety integrity mechanisms (see [Section 6.4.1.5, Built-In Self-Tests \(BIST\)](#)). Dedicated mechanisms are provided to check the availability of safety mechanisms and the functionality of each error reaction path (such as by fake fault injection). The *SPC57EM80xx Safety Manual*, [Table 19 \(LBIST partitions\)](#) and [Table 20 \(Not in LBIST partitions\)](#) contain details about the available LBIST partitions.
- The Fault Collection and Control Unit (see “Fault Collection and Control Unit (FCCU)” chapter for FCCU details) is responsible for collecting and reacting to failure notifications (see [Section 6.4.1.7, FCCU and failure monitoring](#)). MEMU (see [Section 6.4.4.4, Memory Error Management Unit \(MEMU\)](#)) is responsible for the collection and reporting (to the FCCU) of error events in system memories, and flash memory, as well as E2E ECC errors caused by the XBAR, or RAM or flash memory controllers. Error events include ECC corrections and detections as well as faults detected by memory BISTs.
- Common Cause Failures (CCF) are dealt with by a set of measures for both control and avoidance of CCFs spanning system-level approaches (such as temperature and nonfunctional signal monitoring) and back-end techniques (such as isolated silicon areas and routing constraints) (see [Section 6.4.3, Common Cause Failure measures](#)).

6.4.1.1 End-to-End protection on data path

Connections between XBAR masters and slaves (clients) are denoted as data paths. Data corruption on all data paths between the Safety Core and any client is detected with at least 99% coverage via two main safety mechanisms: data from the replicated cores is encoded using Error Correcting Code (ECC), which is implemented with a single-error correction, double-error detection (SEDED) code with a Hamming distance of 4 and includes coverage of addressing information; control signals and address decoding are monitored to verify the data reaches all of the intended clients, from all possible connections to these clients and the intended operation is performed on the target address. [Figure 7](#) shows a general view of ECC schema.

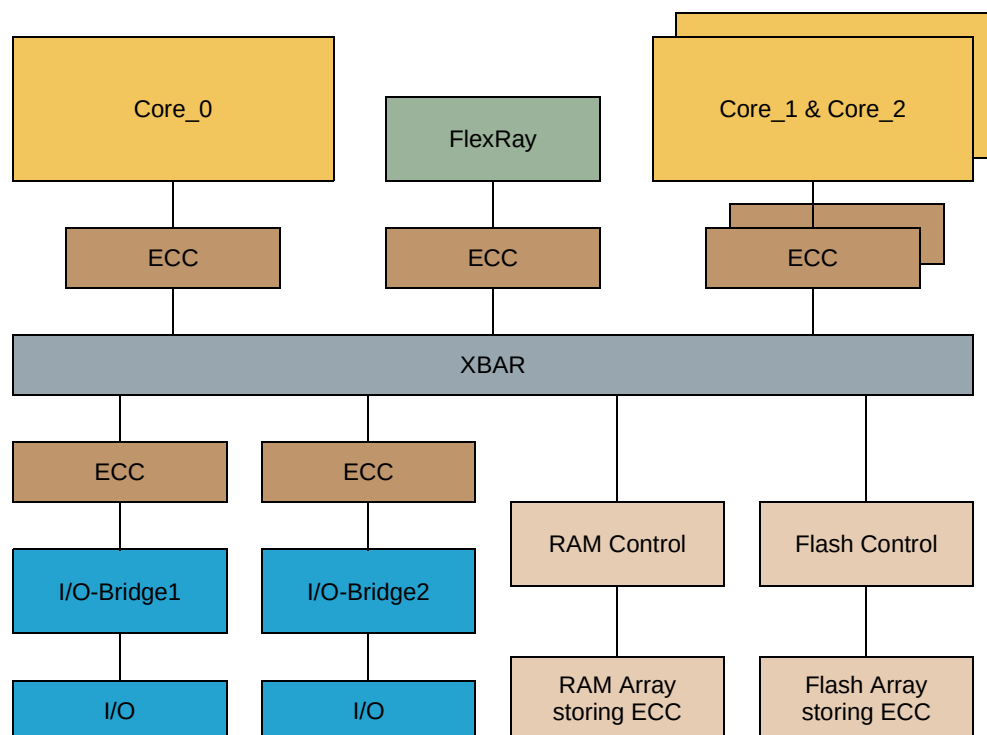


Figure 7. General view of E2E ECC

Note: Specific implementation for the SPC57EM80xx vary depending on the special requirements of RAM and flash memory concerning ECC handling.

ECC bits are generated on writes by XBAR masters (including, but not limited to the Safety Core) and checked on reads. The ECC correction bits are stored alongside the data in flash memory and RAM so, in principle, no ECC logic is necessary at the memories themselves. For this reason the ECC schema is referred to as End-to-End ECC (E2E ECC) in the following sections. For XBAR slaves, apart from memories, new ECC logic is added as these clients cannot store or produce the ECC correction bits. This resolves the problem where ECC needs to be calculated in real time before entering or exiting the ECC-protected data path. This setup leaves the data path downstream of the ECC units (for example, starting at, and downstream of, the peripheral bridges) unprotected by ECC. To detect corruptions introduced on those unprotected data paths, the PBRIDGES and peripherals can be used redundantly (see [Section 6.4.1.3, I/O peripherals](#)), or other mechanisms may be used to validate peripheral operation.

The E2E ECC schema provides high detection capabilities against failures affecting the data content of the transaction. The inclusion of the target address in the computation of the redundancy bits (8 ECC bits) does allow the partial detection of addressing faults as well. To reach the desired integrity level, additional dedicated safety mechanisms are implemented in the data path particularly to:

- Improve the detection capability over addressing failures (no/multiple/wrong address selected), considering faults affecting address transmission (from master to client) as well as the decoding of the address;
- Provide coverage for control failures affecting, for example, the type (read vs. write) or size of a transaction.

Though safety mechanisms protecting the XBAR, the RAM controller, or the flash memory controller are different, they are all based on the feedback of address and control information from the target to the source of the transaction, which is responsible for checking for consistency with respect to the intended transaction. Depending on the portion of the data path covered by the specific safety mechanism, the source can be an XBAR master port rather than the XBAR interface of the RAM or Flash controllers; the target is respectively an XBAR slave port, the RAM array, or the Flash module (see “Crossbar Switch (XBAR)”, “Flash memory controller”, and “RAM controller (PRAM)” chapters).

6.4.1.2 Clock and power monitoring

6.4.1.2.1 Clock

Clocks in the SPC57EM80xx are supervised by Clock Monitor Units (CMUs). The CMUs are driven by the IRCOSC (16 MHz internal oscillator) for independent operation from the monitored clocks. CMUs flag errors associated with conditions due to loss of clock, clock out of a programmable bound (over and under frequency), and loss of reference. If a supervised clock leaves the specified range for the device, a signal is sent to the FCCU.

The clock tree for Core_0 and Core_0 – Checker (the two lockstep cores comprising the Safety Core) consists of separate clock branches that are not shared with any other module. One CMU (CMU_1) is connected to the Main Core clock path, downstream from the common part of the clock tree for the Main and Checker Cores. In this way any frequency failure affecting the Main Core clock or the common clock (SYS_CLK) is detected by the CMU, and any frequency failure affecting the Checker Core clock is detected by the Redundancy Control and Checker Units (RCCUs) (see [27, Clocking](#), for MCU clocking details).

6.4.1.2.2 Power

There are two types of voltage supervisors on the SPC57EM80xx, Low Voltage Detect (LVD) and High Voltage Detect (HVD) monitors. Safety-relevant voltages are supervised for voltages that are out of these ranges. Since safety relevant voltages have the potential to disable the failure indication mechanisms of the MCU (such as FCCU, pads, and so on) the indication of these errors can be used to cause a direct transition of the MCU into the safe state (reset assertion) (see “Power management” and “Power Control Unit (MC_PCU)” chapters for details on power monitoring).

6.4.1.3 I/O peripherals

To allow a safety application to make redundant use of all I/O peripherals, each peripheral has at least two instances (except for the GTM), and each instance is connected to a different PBRIDGE. This means, for example, that if DSPI is provided by the MCU, two DSPI modules (DSPI_n, DSPI_m) are included and connected externally through different pins. Internally, DSPI_n would then be connected to PBRIDGE_A and DSPI_m to PBRIDGE_B and be accessible via different addresses.

The arrangement of I/O peripherals onto two PBRIDGES, as well as further CCF prevention measures, allow redundant use of peripherals while limiting possible causes of CCFs. Redundant usage includes usage of equivalent peripherals in a replicated way as well as usage of functionally different peripherals in, for example, feedback measurement loops. Comparison of redundant operation is the responsibility of the application software, not the safety hardware mechanism. Details on proper I/O peripheral usage can be found in the *SPC57EM80xx Safety Manual*, and the split of peripherals on the two PBRIDGES is shown in [Figure 2 \(Block diagram\)](#) and [Figure 3 \(Periphery allocation\)](#)

6.4.1.4 Communication controllers

Communication controllers provide the ability to exchange information with external components and therefore fall under the same safety reasoning as I/O peripherals. Yet we assume that for high bandwidth communication controllers additional software measures are employed that do not require redundant communication peripherals.

The following communication controllers do not contain special safety mechanisms (above what is included in them by their protocol specifications) nor are they duplicated or spread over the PBRIDGE (in contrast to other I/Os, see [Section 6.4.1.3, I/O peripherals](#)):

- FlexRay
- FEC
- M_CAN
- TT_CAN

Typically, software measures for the communication controllers (also called fault-tolerant communication layer) could contain E2E CRC data protection, sender identification, sequence numbering, and an acknowledgement mechanism.

PSI5 represents an exception to the replication of low bandwidth communications controllers. The module itself is replicated, but the two PSI5 modules share a single synchronization bit needed for staggered sync mode. This single bit is a single source of failures for both PSI5 modules if it triggers spurious synchronizations. The failure mode has to be covered by application level countermeasures, which are in any case needed to cover the single transceiver used in such a configuration.

6.4.1.4.1 Disabling of communication controllers

In the event of a dangerous failure, the MCU offers the capability of disabling transmission of individual channels of communication controllers (such as CAN modules and FlexRay). This prevents the transmission of erroneous messages while preserving the capability of communicating over the diagnostic bus. Disabling outputs is controlled by resetting the SIUL2_MSCRn[SMC] for the pins that are associated with communication controllers where this feature is needed (see the “System Integration Unit Lite2 (SIUL2)” chapter for details on disabling communication signals).

It is understood that FCCU can drive a fault state on FI[0] with either of the following two mechanisms:

- Error pin reaction due to a fault
- Software writing FCCU_CFG[FCCU_SET_CLEAR] = 01b

When the FCCU drives FI[0] to a fault state, the SIUL2 disables the output drivers of all pins whose SIUL2_MSCRn[SMC] = 0. This disables the possible transmission of erroneous messages. As long as the FCCU drives FI[0] to faulty state, the communication controllers' Tx pins are disabled and internal pull-up are enabled. As soon as FI[0] is driven to non-faulty state, the communication controllers' Tx pins are automatically enabled and state of internal pull-up/down restored.

The application should configure SIUL2_MSCRn[SMC] for pins that have active mapping of CAN modules (M_CAN, TT_CAN) or FlexRay functionality and ensure that pin does not stay in undriven state.

Note: The FCCU uses an internal signal to disable the communication controller transmission, so transmission is disabled even when the FCCU cannot drive the FI[0] pin because the pin is

configured as GPIO. Also, if FI[0] is used as error input, externally pulling it down will disable the communication controller transmission if this event drives FCCU to a fault state.

Note: Disabling of communication controllers occurs on the Tx pins of the communication controller (for example, TT_CAN, M_CAN, FlexRay) of the MCU even while the communication controller continues operating. Failed transmissions, when detected by the communication controller, may lead to additional error events.

6.4.1.5 Built-In Self-Tests (BIST)

The term BIST indicates the set of built-in hardware mechanisms that can be used (typically at startup) to avoid the accumulation of latent faults. BIST is a mechanism that permits a device to test itself. On the SPC57EM80xx, BIST is the principal means of meeting the requirement on latent faults as defined by the ISO 26262 standard. Different “flavors” of BIST are implemented in the SPC57EM80xx: LBIST for digital logic, MBIST for memories, and the MCU’s built-in mechanisms for testing analog peripherals. LBIST and MBIST execution are managed by the STCU2, while the testing of analog peripherals requires software intervention to be triggered (see “Self-Test Control Unit (STCU2)” chapter).

6.4.1.5.1 BIST during boot

A device BIST is performed every time the device boots. BIST is performed transparently for the application while the device is still under reset. BIST failure will hold the device in reset, but the BIST will continue running where possible, and inform the system of the failure. Application software can start executing when the BIST finishes successfully without detecting a fault. The boot time BIST comprises:

- Memory BIST for all RAMs and ROM
- Scan-based Logic BIST for digital logic, which is divided into multiple partitions that can be configured to be tested in parallel or sequentially to find the best time versus power consumption trade off.

6.4.1.5.2 Online Logical BIST (LBIST)

Though LBIST and MBIST are primarily intended to be run at startup under STCU2 control, the MCU allows software to trigger a memory self-test or LBIST of one partition during runtime. Since LBIST and MBIST are destructive, a reset is performed for the tested module (memory or partition) before going into operation again. This leads to a reset of the MCU or of those LBIST partitions. If a runtime LBIST requires a reset of the whole device (and not just the LBIST partition), the LBIST result is accessible to software after the reset. If a reset occurs during LBIST, the LBIST will be aborted. [Table 19](#) and [Table 20](#) below show the module groupings for each LBIST partition.

Note: Testing LBIST functionality is ongoing as of this writing. This information is preliminary and will be updated as more data is received.

Table 19. LBIST partitions

LBIST partition	Instance name
LBIST 0 - HSM (also sometimes listed as LBIST 10)	HSM

Table 19. LBIST partitions (continued)

LBIST partition	Instance name
LBIST 1 - IOP Core	IOP core (Core 2)
	IOP instruction cache
	INTC
	STM
	SWT
LBIST 2 - Peripheral Shell	AHB – MPU
	PBRIDGE_A
	PBRIDGE_B
	PIT_0
	PIT_1
	CAN RAM
	CAN
	FCCU
LBIST 3 - Computational Shell	AHB - MPU
	PRAMC
	RAM
	PFLASHC
	Flash memory
	RAM overlay
	AMU
	MEMU
LBIST 4 - Peripheral Shell Masters	eDMA
	DMACHMUX
	DMA TCD RAM
	FEC
	RAM FEC
	FlexRay
	SIPI

Table 19. LBIST partitions (continued)

LBIST partition	Instance name
LBIST 5 - Peripherals	SDADC
	SARADC
	BAM ROM
	CRC
	DSPI
	IIC
	PSI5
	SENT
	REG_PROT
LBIST 6 - GTM	GTM
	GTM RAM
LBIST 7 - Safety Core	Master core (Core 0)
	Instruction cache
	Data cache
	INTC
	STM
	SWT
LBIST 8 - Checker Core	Checker core (Core 0 - Checker)
	RCCU and delay logic
LBIST 9 - Computational Core	Computational Core (Core 1)
	Instruction cache
	Data cache
	INTC
	STM
	SWT

[Table 20](#) shows the modules not covered by the LBIST partitions.

Table 20. Not in LBIST partitions

Functional group	Instance name
System	MC_CGM
	MC_ME
	MC_PCU
	MC_RGM
	SSCM
	WKPU
	SIUL2
	IOMUX
	STCU2
Clocking	PLLs
	CMUs
	XOSC
	IRCOSC
Power	PMC
	ADC bandgap reference
	TSENS
Debug	JTAGC
	SPU
	NEXUS
	LFAST
	DPS
	NAR
	DCI
	JTAGM

6.4.1.6 Redundancy Control and Checker Units (RCCUs)

The RCCU compares a set of equivalent input signals provided by different sources and issues an alarm in case of a mismatch. For example, an RCCU compares the output of the checker core against the output of the main core and issues an alarm if they do not match. In case of a compare mismatch the fault stays stable until it is explicitly cleared.

6.4.1.7 FCCU and failure monitoring

The FCCU offers a hardware mechanism to aggregate error notifications and a configurable means to bring the device to a safe state. No CPU intervention is required for collection and control operation. Error indications are passed from the individual hardware components to the FCCU where the appropriate action is decided (according to the FCCU configuration).

6.4.1.7.1 External error indication

Failure of the MCU is signaled to one or two pins, FI[0] (ERROR0) and FI[1] (ERROR1). FI[0] can also serve as an error input mechanism (see “Fault Collection and Control Unit (FCCU)” chapter, and “FCCU configuration” section of the “Device configuration” chapter, for details on the fault output signals). Both pins can be multiplexed with other functions with different defaults.

The error indication on pins FI[0] and FI[1] is controlled by SIUL2 and FCCU. The mapping of the FCCU output signals on the relevant pins and the associated electrical characteristics are enabled by the SIUL2_MSCR[n] register (see table “SIUL2_MSCR0 - SIUL2_MSCR511 field description” in the “System Integration Unit Lite2 (SIUL2)” chapter for signal mapping details). Once the FCCU is mapped onto the pin, the logic level is driven by the FCCU.

FI[0] is controlled by the FCCU at startup by default, but the state of FI[1] is like other I/O signals (see table “Pin/ball startup and reset states” in chapter “Signal Description” for default states of FI[0] and FI[1]).

Since it is possible to reset FCCU and SIUL2 independently, the behavior of FI[0] and FI[1] is the result of the combined state of SIUL2_MSCR[n] and the FCCU. (see section “Reset interface” in chapter “Fault Collection and Control Unit (FCCU)” for FCCU reset details, and chapter “Reset and Boot” for the state of the FCCU relative to the system during different reset phases).

The FCCU_STAT[ESTAT] error status flag can be read to show whether the FCCU is in an error state. This flag can be written by software to either a 1 (fault) or 0 (operational) when the FCCU is in operational state. Another flag, FCCU_STAT[PSTAT], is accessible through the register interface (which mirrors the physical state of the *Fn* external pin's value, though this may differ from the logical state if a toggling protocol is used).

6.4.1.7.2 Failure handling

The FCCU is an autonomous module that is responsible for reacting to failure indicators. A different reaction can be configured for each failure source. Overall failure reaction time requires time for detecting, processing, and indicating the error. During this time, the SPC57EM80xx could provide wrong results to the system.

Failure sources include:

- All failure indication signals from modules within the MCU
- Control logic and signals monitored by the FCCU itself (see chapter “Fault Collection and Control Unit (FCCU)”)
- Software-initiated failure indications. For example, software signals the FCCU that it has evidence of a failure. Keep in mind that software can also directly influence the state of the *Fn* pins.
- External failure input.

Available failure reactions are:

- Assertion of an interrupt (maskable or non-maskable)
- Resetting the MCU
- Changing the state of the failure indication pins, ERROR_n
- Disabling the transmission capabilities of communication controllers (FlexRay, CAN, FEC) (Note: only possible in conjunction with changing the state of the failure indication pins)
- No reaction

Software can read the failure source that caused a fault, either before or after a functional reset (the condition indicators are not volatile). Software can also reset the failure, but the external failure indication will stay in failure mode for a configurable minimum time. If necessary, software can also reset the MCU.



6.4.1.7.3 FCCU failure inputs from other modules

[Table 35](#) shows the different failure input signals of the FCCU, how they can be tested, and their default configuration after reset.

Table 21. FCCU failure inputs⁽¹⁾

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
0	TEMP_ERROR	Event indication from temperature sensor. The fault is cleared by clearing the status in PMC_dig_EPR_TD, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	PMC_dig_EPR_TD Section 60.3.1.44
1	LVD_ERROR	Voltage out of range indication from LVDs. The fault is cleared by clearing the status in PMC_dig_EPR_VDn, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LVD assertion test by tap point manipulation	No reaction	PMC_dig_EPR_VD3 PMC_dig_EPR_VD4 PMC_dig_EPR_VD9 PMC_dig_EPR_VD10 PMC_dig_EPR_VD13 PMC_dig_EPR_VD14 Section 60.3, Memory Map/Register Definition
2	HVD_ERROR	Voltage out of range indication from HVDs. The fault is cleared by clearing the status in PMC_dig_EPR_VDn, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Self-test by TAP point manipulation	No reaction	PMC_dig_EPR_VD7 PMC_dig_EPR_VD8 PMC_dig_EPR_VD12 Section 60.3, Memory Map/Register Definition
3	DPMC_DCF_SAFETY_ERR	Digital PMC initialization error during DCF data load. The fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]. The status gets cleared if the fault is not persistent.	LBIST	No reaction	Section 60.9

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
4	MEMORY_DCF_SAFETY_ERR	Indication of fault during timing/repair configuration initialization by SSCM of different memories. The status is cleared by clearing SSCM_STATUS[CER], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]. The status gets cleared if the fault is not persistent. Additionally, this error will cause a MBIST failure.	LBIST	No reaction	Section 59.3.1.1
5	SSCM_XFER_ERR	SSCM transfer error (during the SSCM and STCU DCF loading). The fault is cleared by clearing SSCM_STATUS[CER], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Not testable (but wrong configuration can trigger other detection)	No reaction	Section 59.3.1.1
6	STCU_UF	STCU2 unrecoverable fault indication during self-test. It is simultaneously reported to the MC_RGM which will initiate a destructive reset, if enabled. In the STCU2, self-test failures can be configured as recoverable or unrecoverable faults which would then lead to assertion of the corresponding fault line to the FCCU. Fault is cleared by clearing STCU2_ERR_STAT[UFSF] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	SW check of BIST MISR or Feedback loop via FCCU Fake bit	Long Reset	Section 82.5.11 Section 57.1.3
7	STCU_RF	STCU2 recoverable fault indication during self-test. In the STCU2, self-test failures can be configured as recoverable or unrecoverable faults which would then lead to assertion of the corresponding fault line to the FCCU. Fault is cleared by clearing STCU2_ERR_STAT[RFSF] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	SW check of BIST MISR or Feedback loop via FCCU Fake bit	No reaction	Section 82.5.11

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
8	STCU_LMBIST_USR_ERR	Activation of LBIST or MBIST control during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]. The status will clear if the fault is not persistent.	Feedback loop via FCCU Fake bit.	No reaction	—
9	SPURIOUS_DEBUG_SSCM_ACTIVATION	JTAG, NPC or Debug functionality moved out of reset or SSCM activation during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Feedback loop via FCCU Fake bit	No reaction	—
10	RCCU_0	Safety cores out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Feedback loop via FCCU Fake bit connected to additional RCCU input	No reaction	Section 6.4.1.6
11	RCCU_1	Safety cores out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Feedback loop via FCCU Fake bit connected to additional RCCU input	No reaction	Section 6.4.1.6
12	SWT2_ResetReq	Reset request from Software Watchdog of Peripheral Core_2 domain. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT2.	LBIST + SW test by not triggering SWT during startup	Long Reset	Chapter 44: Software Watchdog Timer (SWT)
13	SWT1_ResetReq	Reset request from Software Watchdog of Main Core_1 domain. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT1.	LBIST + SW test by not triggering SWT during startup	No reaction	Chapter 44: Software Watchdog Timer (SWT)

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
14	SWT0_InterrupReq	First timeout interrupt request from Software Watchdog of Safety Core (Main Core_0). Fault is cleared by clearing SWT0_SWT_IR[TIF], followed by clearing the FCCU channel status FCCU_RF_Sn[RFSm].	LBIST + SW test by not triggering SWT during startup	No reaction	Chapter 44: Software Watchdog Timer (SWT)
15	SWT0_ResetReq	Second timeout (reset) request from Software Watchdog of Safety Core (Main Core_0). The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT0.	LBIST + SW test by not triggering SWT during startup	No reaction	Chapter 44: Software Watchdog Timer (SWT)
16	MEMU_RAM_CE	Indication by MEMU of occurrence of correctable errors in system RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[SR_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Injection of RAM ECC error via ECC bit access and later reading	No reaction	Section 79.6.2.2
17	MEMU_RAM_UCE	Indication by MEMU of occurrence of uncorrectable errors in system RAMs. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Injection of RAM ECC error via ECC bit access and later reading	No reaction	Section 79.6.2.2
18	MEMU_RAM_BOV ⁽³⁾	Indication of overflow of system RAM ECC buffer in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Feedback loop via FCCU Fake bit and MEMU output	No reaction	Section 79.6.2.2

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
19	MEMU_PER_CE ⁽³⁾	Indication by MEMU of the occurrence of correctable errors in peripheral RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[PR_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	LBIST + Feedback loop via FCCU Fake bit and MEMU output	No reaction	Section 79.6.2.2
20	MEMU_PER_UCE ⁽³⁾	Indication by MEMU of the occurrence of uncorrectable errors in peripheral RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[PR_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	LBIST + Feedback loop via FCCU Fake bit and MEMU output	No reaction	Section 79.6.2.2
21	MEMU_PER_BOV ⁽³⁾	Indication of overflow of system RAM ECC buffer in MEMU. The fault is cleared by clearing MEMU_ERR_FLAG[PR_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	LBIST + Feedback loop via FCCU Fake bit and MEMU output	No reaction	Section 79.6.2.2
22	MEMU_FLS_CE	Indication by MEMU of occurrence of correctable errors in flash memory. The fault is cleared by the clearing MEMU_ERR_FLAG[F_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Reading preprogrammed correctable ECC error from UTEST flash memory	No reaction	Section 79.6.2.2
23	MEMU_FLS_UCE	Indication by MEMU of occurrence of uncorrectable errors in flash memory. The fault is cleared by the clearing MEMU_ERR_FLAG[F_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Reading preprogrammed uncorrectable ECC error from UTEST flash memory	No reaction	Section 79.6.2.2

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
24	MEMU_FLS_BOV ⁽³⁾	Indication of overflow of flash memory ECC buffer in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[F_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Feedback loop via FCCU Fake bit and MEMU output	No reaction	Section 79.6.2.2
25	IMA_SOC	MCU IMA active	LBIST + IMA read	No reaction	—
26	IMA_HSM	HSM IMA active	LBIST + IMA write	No reaction	—
27	SARB_WDG	Reference-SAR ADC watchdog is triggered. Fault is cleared by clearing the asserted bits [WDGnL, WDGnH] in SARB_WTISR, followed by clearing FCCU_RF_Sn[RFSm].	LBIST + ADC conversion outside the watchdog range	No reaction	Section 40.4.7
28	SARn_WDG	OR'd outputs of watchdogs, SAR_ADC[0:7]. Fault is cleared by clearing any asserted bits [WDGnL, WDGnH] in WTISR register of any functional SAR ADC, followed by clearing FCCU_RF_Sn[RFSm].	LBIST + ADC conversion outside the watchdog range	No reaction	Section 40.4.7
29	FM_PLL_0	PLL0 loss of lock. In order for this fault source to be asserted in the event of loss of lock, the Loss of Lock Interrupt needs to be enabled by writing (PLLDIG_PLL0CR[LOLIE]) = 1. Fault is cleared by clearing PLLDIG_PLL0SR[LOLF], followed by clearing FCCU_RF_Sn[RFSm].	LBIST + Forced loss by PLL reconfiguration	No reaction	—
30	FM_PLL_1	PLL1 loss of lock. In order for this fault source to be asserted, the Loss of Lock Interrupt needs to be enabled by writing PLLDIG_PLL1CR[LOLIE] = 1. Fault is cleared by clearing PLLDIG_PLL1SR[LOLF], followed by clearing FCCU_RF_Sn[RFSm].	LBIST + Forced loss by PLL reconfiguration	No reaction	—

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
31	CMU_0_OSC ^{(4),(5)}	Loss of XOSC clock. The fault is cleared by clearing CMU_ISR[OLRI] of CMU_0, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	Section 29.4.1.5
32	CMU_0_PLL ^{(4),(6)}	SYSCLK frequency out of range. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_0, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	LBIST + Test by misconfiguration ⁽⁶⁾	No reaction	Section 29.4.1.5
33	CMU_comp_subsys ^{(4),(7)}	Exception from monitoring internal clocks of comp_subsys. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_n, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	LBIST + Test by misconfiguration ⁽⁶⁾	No reaction	Section 29.4.1.5
34	CMU_other ^{(4),(8)}	Exception from monitoring internal clocks of other subsystems. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_n, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	LBIST + Test by misconfiguration ⁽⁶⁾	No reaction	Section 29.4.1.5
35	PERIPH XBAR Transaction Monitor	Indication of a hardware fault resulting in corrupted transaction through the peripheral XBAR. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	Chapter 18: Crossbar Integrity Checker (XBIC)
36	EDC_ECC_FLASH	Indication of a hardware fault in the ECC of the flash memory resulting in corrupted ECC detection/correction. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
37	EDC_ECC_FLASH_C	Indication of hardware fault in the flash memory controller resulting in corrupted ECC detection / correction. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
38	ENC_ERR_FLASH	Indication of hardware fault resulting in corrupted flash memory access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
39	ADDR_FDBK_ERR_FLASH_C	Indication of an addressing error in the flash memory controller resulting in corrupted flash memory access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
40	COMP XBAR Transaction Monitor	Indication of an addressing or control fault resulting in corrupted transaction through the computational XBAR. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	Chapter 18: Crossbar Integrity Checker (XBIC)
41	ON_CAL_RAM_ERR	Indication of an addressing error in the on-chip overlay RAM resulting in corrupted access to on-chip overlay RAM. Status should be cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
42	SAFE_CAL_ERR	Indication of hardware fault in the flash memory controller resulting in corrupted overlay remap evaluation. Status should be cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
43	ADDR_FDBK_ERR_PRAM + RAM_LWB_ERR	Indication of an addressing error in system RAM or a write error in the RAM controller resulting in corrupted RAM access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
44	MPU_ERROR	Indication of the SMPU that it prevented an unauthorized access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
45	EDC_ERR_PRAM	Indication of a fault in the RMW logic of the system RAM resulting in an incorrect RMW access and a potentially corrupted RAM word. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Feedback loop via FCCU Fake bit	No reaction	—
46	DFT1	Test circuitry Group 1 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
47	DFT2	Test circuitry Group 2 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
48	DFT3	Test circuitry Group 3 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—
49	DFT4	Test circuitry Group 4 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	—

Table 21. FCCU failure inputs⁽¹⁾ (continued)

FCCU channel (2)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information
50	Safety Core Machine Check Exception ⁽⁹⁾	Safety Core exception indication. The fault is cleared by clearing the status in MCSR register of the Safety core followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Willfully cause an appropriate exception (for example, access a non-existing address)	No reaction	—
51	Lockstep mode	Indication of disabled Checker Core or RCCUs. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + Feedback loop via FCCU Fake bit	No reaction	Section 6.4.2
52	Safe_Mode	Indication of safe mode request assertion from MC_RGM. The event is also captured in MC_ME and available as status for crosscheck. Status should be cleared first in the MC_RGM_FES, then followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST + (in principle requesting an RGM "safe mode" though that is not typical, and should be avoided)	No reaction	Section 57.3.1.5
53	Compensation Disable	Pad compensation disabled fault which potentially results in distorted transmissions on various I/Os. The indication is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	—	No reaction	—
54	EIN_ERR ⁽¹⁰⁾	Error input pin (FI[0], external of the MCU) signalling an error condition. The fault is cleared by clearing the fault driving FI[0] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	External activation of EIN. External entity has to pull the error input low.	No reaction	—
55	SWT3_ResetReq	Reset request from Security Watchdog. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared in the SWT3.	LBIST+ SW test by not triggering SWT during startup	No Reaction	Section 7.7.2.1

1. All of these faults require a long functional reset.

2. See "Fault Collection and Control Unit (FCCU)" chapter for channel mapping details.

3. Error reaction path check contents subject to change.
4. See Clock generation figure ([Figure 258 \(Clock generation \(part 1 of 2\)\)](#)) in the Clocking chapter ([27, Clocking](#)).
5. CMU_0 OLR event + XTAL pin floating + EXTAL Pin Floating.
6. CMU_0 FLL and FHH events.
7. CMU_1, CMU_2, CMU_3, FLL, and FHH events.
8. CMU_4, CMU_5, CMU_6, CMU_7, CMU_8, CMU_9, CMU_10, CMU_11, FLL and FHH events.
9. See “Exceptions and Conditions” table in the *e200z4 Core Reference Manual* for all exceptions which can occur in the core and table “Error report machine check exceptions” for errors indicated by a machine check exception. Note that the following do not cause a machine check and thus do not signal to the FCCU: access blocked by MPU, wrongly aligned access, execution of a privileged or trap instruction, FPU exceptions. Such errors will have to be handled by the respective exception handler if they are safety-relevant.
10. EIN_ERR can be used only when bistable protocol is selected for external error indication, and the FI[0] pin is configured as open drain.

6.4.1.8 Operational interference protection

Being a multi-master system, SPC57EM80xx provides safety mechanisms to prevent non-safety masters from interfering with the operation of the Safety Core, as well as mechanisms to handle the concurrent execution of software with different (lower) ASIL. Interference freedom is guaranteed via a hierarchical memory protection schema including:

- MPUs
- PBRIDGES
- Register protection.

There are two Memory Protection Unit levels included in the SPC57EM80xx. The Core Memory Protection Unit (CMPU) is a mechanism included in each core (except HSM core) to protect address ranges against access by software developed according to lower ASIL. The CMPU is typically used by the operating system to ensure inter-task interference protection.

The second MPU level is provided by the System MPUs (SMPU) located in each XBAR. The MPUs prevent access of different bus masters to address ranges and are typically used by the safety application to prevent NoSaMos (such as the DMA or the Performance Core) to access the application's safety-relevant resources.

Furthermore, the PBRIDGE can restrict read and write access to individual I/O modules based on the origin of the access and its state (user mode/supervisor mode).

Finally, the register protection included allows individual register to be locked against any manipulation without unlocking.

These safety mechanisms are further described in the "System Memory Protection Unit (SMPU)" chapter.

6.4.1.9 FCCU supervision (FOSU)

As the FCCU is a central component in reacting to errors, it is itself supervised even though an error in it can only cause a latent failure. This supervision is provided by the FCCU Output Supervision Unit (FOSU). The FOSU receives failure indications at the same time as the FCCU. Unless the respective failure is switched off, the FOSU will observe the outputs of the FCCU (IRQ, RESET, F_n). If the FCCU does not react within a predefined interval as specified in the "FCCU configuration" section of the "Device configuration" chapter, on one of those outputs to the incoming failure indication, the FOSU assumes the FCCU has failed, and causes a destructive reset.

The FOSU does not require any configuration by software.

6.4.2 Dual-core lockstep

The SPC57EM80xx duplicates its safety-relevant processing elements and compares their operation in Lockstep mode (LSM). This Safety core consists of two cores, Checker Core_0 and Main Core_0, and as far as software is concerned they behave as one core.

Main Core_0 is the main execution core of the pair, where Checker Core_0 follows the execution of the Main Core in lockstep.

The replicated processing elements contain:

- Core
- Cache control
- Local memory control
- Core MPU
- Core System Bus Interface, including e2eECC logic

Together each set of replicated elements forms a channel (for example, the Main channel and the Checker channel).

Elements that are not replicated are:

- Performance or I/O cores
- Cache and local memory arrays
- INTC
- Interconnect (XBAR)
- RAM controller
- Flash memory controller
- eDMA controller

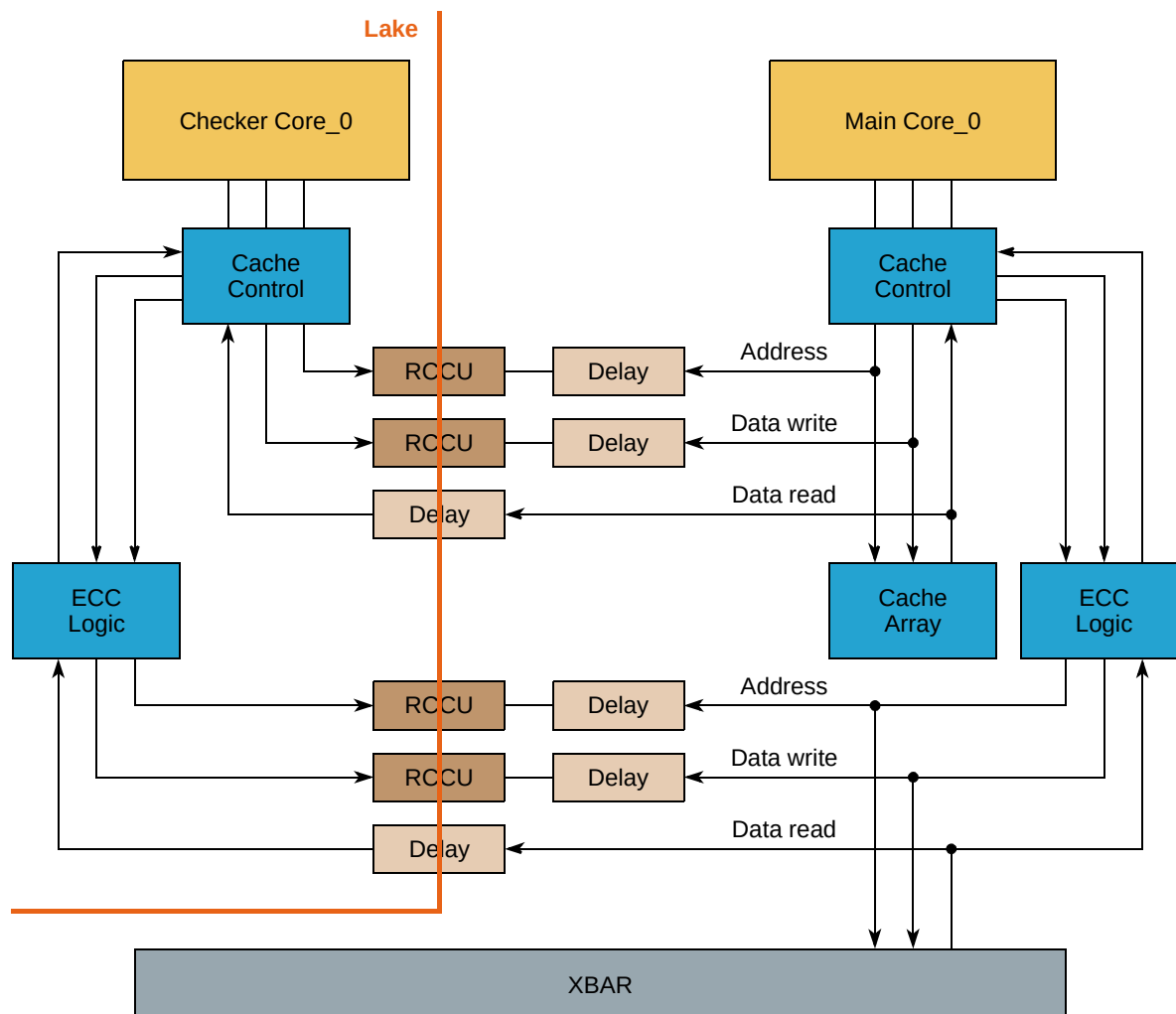
Note: This is not a complete list of all non-replicated elements. But the elements listed are sometimes replicated in similar lockstep architectures.

Equivalent operation of replicated resources is supervised by comparators on all functional signals leaving the channels for the rest of the MCU. Any operational deviations between the supervised signals will cause the FCCU to be notified of the discrepancy.

The Checker Core does not have a direct connection to the XBAR. All of the outputs of Checker Core_0 that target the XBAR (as well as any other non-duplicated resource, like local memories) will end in an RCCU for verification, and all the inputs to Checker Core_0 from the XBAR will be split off from the Main Core_0 XBAR inputs.

An abstract view of the implementation including delays and RCCUs is shown in [Figure 8](#), but does not show local memory.

Figure 8. Dual core concept



A delay of two clock cycles exists between the Main and Checker channel to reduce their susceptibility to CCFs caused by power and clock disturbances. Outputs of the Main Core (such as XBAR addresses) are registered and delayed before being forwarded to the RCCUs, while inputs signals (such as XBAR read data) are registered and delayed before being fed to the Checker channel.

LSM can be disabled during boot by reprogramming the flash memory or on-line by disabling the checker core but not the master core^(d). LSM is also automatically disabled when Core_0 enters debug mode and automatically reenabled when it leaves it. If LSM is disabled, the RCCUs are disabled, and a signal is sent to the FCCU. If SW enables the checker core by writing in MC_ME - MC_ME_CCTL register while LSM is disabled (not recommended), this does not enable LSM and the RCCUs will not generate any error, in case of mismatch. Thus no user-initiated dynamic enabling of LSM is possible. A reset is needed to reestablish LSM. Also, in contrast to the SPC560ELx, no Decoupled Parallel Mode (DPM) is available (Main Core_0 and Checker Core_0 cannot operate as two

d. Disabling both cores in a single mode transition will not disable LSM.

independent cores running different software). Main Core_0 is able to operate as an independent core if Checker Core_0 is disabled. The status of LSM for the Safety core can be verified by checking the appropriate FCCU register.

6.4.3 Common Cause Failure measures

Various measures are included to prevent CCFs from endangering the effectiveness of the replication of the Safety Core or of the peripherals. These measures include physical separation of the components on the die, routing restrictions and supervision of clock, power, temperature, test and debug signals. In general these measures are independent from the software.

There are also configuration registers that, on change, can affect the execution of the MCU's safety function and, simultaneously, disable the respective safety mechanism. These are either protected against bit flips or those flips are detected by independent measures. These same registers are also protected against accidental software writes by employing the register protection described above.

6.4.4 ECC

Error correcting codes are used for end-to-end protection from cores to system storage as well as for individual protection of peripheral RAMs.

6.4.4.1 ECC for storage

All storage (system RAM, local memory, flash memory, peripheral RAM) used in normal operation is protected by ECC with SECDED (Single Error Correct and Double Error Detect). Exceptions to this are noted below:

- The caches in the cores are protected by EDC (Error Detection Code), only detecting, but not correcting errors.
- FlexRay's schedule information RAM is protected by EDC only.
- The HSM's AES program extension RAM are not protected by ECC.

A list showing the implementation of RAMs with ECC (including address protection) and for the SPC57EM80xx is shown in [Table 22](#).

Table 22. ECC RAM implementations

Location	Memory	Memory classification	ECC	Address in ECC	Reacts to uncorrectable error
Core_0 (Safety Core) (Master and Checker cores)	I-Mem	System RAM	SEC/DED	Y	Y
	D-Mem	System RAM	SEC/DED	Y	Y
	I-Cache	System RAM	EDC	Y	Y
	D-Cache	System RAM	EDC	Y	Y
	ITAG	System RAM	SEC/DED	Y	Y
	DTAG	System RAM	SEC/DED	Y	Y

Table 22. ECC RAM implementations (continued)

Location	Memory	Memory classification	ECC	Address in ECC	Reacts to uncorrectable error
Core_1	I-Mem	System RAM	SEC/DED	Y	Y
	D-Mem	System RAM	SEC/DED	Y	Y
	I-Cache	System RAM	EDC	Y	Y
	D-Cache	System RAM	EDC	Y	Y
	ITAG	System RAM	SEC/DED	Y	Y
	DTAG	System RAM	SEC/DED	Y	Y
Core_2	I-Mem	System RAM	SEC/DED	Y	Y
	D-Mem	System RAM	SEC/DED	Y	Y
	I-Cache	System RAM	EDC	Y	Y
	ITAG	System RAM	SEC/DED	Y	Y
HSM	PRAM	Other	SEC/DED	Y	N
	I-Cache	Other	SEC/DED	Y	N
	ITAG	Other	SEC/DED	Y	N
	C3	Other	No	N	N
FlexRay	DRAM	Peripheral	SEC/DED	N	Y
	LRAM	Peripheral	EDC	N	N
GTM	FIFO	Peripheral	SEC/DED	Y	N
	(MCSi)-RAM0, i=0~3	Peripheral	SEC/DED	Y	N
	(MCSi)-RAM1, i=0~3	Peripheral	SEC/DED	Y	N
	DPLL-1A	Peripheral	SEC/DED	Y	N
	DPLL-1B	Peripheral	SEC/DED	Y	N
	DPLL-2	Peripheral	SEC/DED	Y	N
TTCAN	TTCAN	Peripheral	SEC/DED	N	Y
NAR	NAR	Other	No	—	N
Platform/DMA	DMA	Peripheral	SEC/DED	N	Y
Platform	Overlay	System RAM	SEC/DED	Y	Y
Platform	SRAM	System RAM	E2E-ECC	Y	Y
Platform	AMU	Peripheral	SEC/DED	N	N
Platform	FEC/FIFO	Peripheral	SEC/DED	N	N
Platform	FEC/MIB	Peripheral	SEC/DED	N	N

As shown in [Table 22](#), some memories, particularly system storage, use an ECC computed over data and address to detect data and addressing faults (no/wrong/multiple selection). In addition, these same memories include dedicated measures against addressing and control faults (such as address/control feedback). This is different for storage related to PeMos.

PeMos, in general, use an ECC without address error protections and do not include additional measures to detect them. All ECC calculations for RAMs in PeMos are accomplished by logic outside of the RAM, not in the RAM block.

An exception to this is the GTM's RAMs. They include addressing failure detection measures in the ECC. In addition, the individual RAM ports of the GTM do not share common parts and connect to individual (non-interfering) RAM arrays.

6.4.4.2 All-x words and ECC

There is a special case for legal ECC values in the SPC57EM80xx. Memory entries that are all zeros (All-0) or all ones (All-1), including the ECC parity bits, are not legal for memory that is checked by ECC. The flash memory allows All-1, corresponding to the status of an erase block, as a valid codeword.

Memories that include addresses in the ECC calculation do not specifically protect against All-0 or All-1. This means that for some addresses All-0 or All-1 may be legal.

All-0 and All-1 memory content is indicated in different ways. For memories that do not include address into the ECC calculation, All-0 and All-1 will be uncorrectable errors. For all memories that include address into the ECC code-bit calculation, since the ECC checkbits depend on the address, it is not possible to generate an uncorrectable error indication for all the possible addresses. Therefore, an All-x content may result in a correctable error.

Note: For flash memory, additional dedicated safety mechanisms exist to detect failures that have the potential of leading to an All-1 word (see "Embedded Flash Memory" chapter, for more details on flash memory safety mechanisms).

6.4.4.3 ECC failure handling

Single-bit and double-bit errors (correctable and uncorrectable errors) are signaled to the FCCU unless filtered by the MEMU. The MEMU (see [Section 6.4.4.4, Memory Error Management Unit \(MEMU\)](#)) may filter ECC error notification for known ECC error addresses (known permanent correctable errors). Actual implementation will signal errors, not to the FCCU, but to the MEMU, which filters and then forwards unfiltered notifications in an aggregated manner to the FCCU.

6.4.4.4 Memory Error Management Unit (MEMU)

The MEMU is responsible for the collection and reporting of error events associated with ECC logic used on SRAM (system RAMs in this context are those RAMs with "System RAM" memory classification as in [Table 22](#)), peripheral RAM, and flash memory. When ECC error events occur, the MEMU receives an error signal that causes an event to be recorded, and possibly sets corresponding error flags that are reported to the FCCU.

The MEMU stores the addresses of ECC errors that have occurred in a table as follows:

- Uniqueness of the addresses in the table is ensured. For example, if an address is already stored in the MEMU's table, correctable errors at that address will no longer be reported to the FCCU.
- Software can read and modify the MEMU table and remove individual entries (by marking them as invalid). For example, this allows software to invalidate a new entry in the MEMU table and wait for a repeat occurrence of the ECC error indicating a permanent error in memory.
- If the MEMU table overflows, an additional signal (instead of the ECC error signal) is sent to the FCCU.

As the MEMU aggregates address information from several sources, it might not be able to process all simultaneously arriving reports. This is called a simultaneous overflow and is signaled to the FCCU as a buffer overflow (see “Memory Error Management Unit (MEMU)” chapter for more information).

6.4.4.4.1 Interface to ECC units

The MEMU receives data according to the following interface per ECC unit connected:

- Whether the error is a Single-Bit-Error (SBE) or Uncorrectable Error (UCE) type
- Address of the memory where the error occurred
- A configuration specifying whether the ECC unit is attached to a vital memory (SRAM, local memory, and system RAM of Safety Core), to flash memory, or to a peripheral RAM

If an error is signaled, it is compared against all errors known for that storage:

Note: If a previously known error has been marked invalid by software, no comparison against the address will occur.

- If no entry of that address is already in the buffer, it is to be added into the appropriate table depending on the SBE versus UCE indicator

Note: For this comparison the bit position will not be taken into account. This is in contrast to MBIST reporting.

- If there is no free entry left in that buffer, the overflow flag is set.

If a valid entry of that address is in the buffer, the following occurs:

- If the entry indicates an SBE in that address, and the error indicated is uncorrectable, a new entry is added in the MEMU buffer.
- In all other cases, nothing is changed in the entry.

7 Device Configuration

7.1 Introduction

This chapter provides details on the individual modules of the microcontroller. It includes:

- Module block diagrams showing immediate connections within the device
- Specific module-to-module interactions not necessarily discussed in the individual Module chapters
- Links for more information

7.2 Core modules

The SPC57EM80xx has cores in three distinct modules:- HSM (described in the HSM chapter of the *SPC57EM80xx Microcontroller Security Reference Manual*); GTM (described in the *GTM Reference Manual*; and the main platform (described below).

The main platform has four separate cores that perform various computational and control functions

- The main computational shell consists of Main Core_0 and Main Core_1, both of which use an e200z420n3 core. These two cores are used for general computational functions.
- A third core, referred to as Checker Core_0s, uses an e200z419 core which is a true subset of the e200z420n3 core. When enabled, Checker Core_0s operates in lock step mode with Main Core_0 executing exactly the same instructions as Main Core_0. Thus, Checker Core_0s checks to insure that Main Core_0 is executing correctly. There is no lock step function associated with Main Core_1.
- The fourth processor, Peripheral Core_2, employs an e200z425n3 core and is generally used to control various I/O modules.

For more information, please see [Chapter 13: Core Complex Overview](#).

7.2.1 Core Reset Settings

[Table 23](#) shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately following a system reset.

Table 23. Reset settings for e200z4 resources

Resource	System reset setting
Program Counter	p_rstbase[0:29] 2'b00 ⁽¹⁾
GPRs	Unaffected ⁽²⁾
CR	Unaffected ⁽²⁾
BUCSR	0x0000_0000
CSRR0	Unaffected ⁽²⁾
CSRR1	Unaffected ⁽²⁾
CTR	Unaffected ⁽²⁾

Table 23. Reset settings for e200z4 resources (continued)

Resource	System reset setting
DAC1	0x0000_0000 ⁽³⁾
DAC2	0x0000_0000 ⁽³⁾
DAC3	0x0000_0000 ⁽³⁾
DAC4	0x0000_0000 ⁽³⁾
DBCR0	0x0000_0000 ⁽³⁾
DBCR1	0x0000_0000 ⁽³⁾
DBCR2	0x0000_0000 ⁽³⁾
DBCR4	0x0000_0000 ⁽³⁾
DBCR5	0x0000_0000 ⁽³⁾
DBCR6	0x0000_0000 ⁽³⁾
DBCR7	0x0000_0000 ⁽³⁾
DBCR8	0x0000_0000 ⁽³⁾
DBSR	0x1000_0000 ⁽³⁾
DDAM	0x0000_0000 ⁽³⁾
DDEAR	Unaffected ⁽²⁾
DEAR	Unaffected ⁽²⁾
DEVENT	0x0000_0000 ³
DSRR0	Unaffected ⁽²⁾
DSRR1	Unaffected ⁽²⁾
DVC1	Unaffected ⁽²⁾
DVC2	Unaffected ⁽²⁾
ESR	0x0000_0000
HID0	0x0000_0000
HID1	0x0000_0000
IAC1	0x0000_0000 ⁽³⁾
IAC2	0x0000_0000 ⁽³⁾
IAC3	0x0000_0000 ⁽³⁾
IAC4	0x0000_0000 ⁽³⁾
IAC5	0x0000_0000 ⁽³⁾
IAC6	0x0000_0000 ⁽³⁾
IAC7	0x0000_0000 ⁽³⁾
IAC8	0x0000_0000 ⁽³⁾
IVPR	Unaffected ⁽²⁾
LR	Unaffected ⁽²⁾
L1CFG0, L1CFG1 ⁽⁴⁾	—

Table 23. Reset settings for e200z4 resources (continued)

Resource	System reset setting
L1CSR0, 1	0x0000_0000
L1FINV0, 1	0x0000_0000
MAS1	Unaffected ⁽²⁾
MAS2	Unaffected ⁽²⁾
MAS3	Unaffected ⁽²⁾
MAS4	Unaffected ⁽²⁾
MCAR	Unaffected ⁽²⁾
MCSR	0x0000_0000
MCSRR0	Unaffected ⁽²⁾
MCSRR1	Unaffected ⁽²⁾
MMUCFG ⁽⁴⁾	—
MPU0CSR0	0x0000_0000
MPU0CFG ⁽⁴⁾	—
MSR	0x0000_0000
NPIDR	0x0000_0000
PID0	0x0000_0000
PIR	0x0000_00 p_cpuid[0:7]1
PVR ⁽⁴⁾	—
SPEFSCR	0x0000_0000
SPRG0	Unaffected ⁽²⁾
SPRG1	Unaffected ⁽²⁾
SPRG2	Unaffected ⁽²⁾
SPRG3	Unaffected ⁽²⁾
SRR0	Unaffected ⁽²⁾
SRR1	Unaffected ⁽²⁾
SVR ⁽⁴⁾	—
XER	0x0000_0000

- The levels that determine these bits are set by the factory and may change between revisions of the device.
Main_Core_0: p_cpuid[0:7] = 0x00
Main_core_1: p_cpuid[0:7] = 0x01
Peripheral_Core_2: p_cpuid[0:7] = 0x02
- Undefined on **POR** assertion, unchanged on **external PORESET** assertion.
- Reset by processor reset **external PORESET** if DBCR0[EDM]=0, as well as unconditionally by **POR**.
- Read-only registers.

7.2.2 Special Purpose Register Summary

PowerISA 2.06 and implementation-specific SPRs for the e200z420n3, e200z419, and e200z425n3 cores are listed in [Table 24](#). All registers are 32 bits in size. Register bits are numbered from bit 0 to bit 31 (most-significant to least-significant). Shaded entries represent optional registers. An SPR register may be read or written with the **mfscr** and **mtscr** instructions. In the instruction syntax, compilers should recognize the mnemonic name given in [Table 24](#).

Table 24. Special Purpose Registers (SPRs)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
XER	Integer Exception Register	1	R/W	No	No
LR	Link Register	8	R/W	No	No
CTR	Count Register	9	R/W	No	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No
SRR1	Save/Restore Register 1	27	R/W	Yes	No
PID0	Process ID Register	48	R/W	Yes	No
CSRR0	Critical Save/Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/Restore Register 1	59	R/W	Yes	No
DEAR	Data Exception Address Register	61	R/W	Yes	No
ESR	Exception Syndrome Register	62	R/W	Yes	No
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
SPRG0	SPR General 0	272	R/W	Yes	No
SPRG1	SPR General 1	273	R/W	Yes	No
SPRG2	SPR General 2	274	R/W	Yes	No
SPRG3	SPR General 3	275	R/W	Yes	No
PIR	Processor ID Register	286	R/W	Yes	No
PVR	Processor Version Register	287	Read-only	Yes	No
DBSR	Debug Status Register	304	Read/Clear ⁽¹⁾	Yes	No
DBCR0	Debug Control Register 0	308	R/W	Yes	No
DBCR1	Debug Control Register 1	309	R/W	Yes	No
DBCR2	Debug Control Register 2	310	R/W	Yes	No
IAC1	Instruction Address Compare 1	312	R/W	Yes	No
IAC2	Instruction Address Compare 2	313	R/W	Yes	No
IAC3	Instruction Address Compare 3	314	R/W	Yes	No
IAC4	Instruction Address Compare 4	315	R/W	Yes	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	Yes	No

Table 24. Special Purpose Registers (SPRs) (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
DVC1	Data Value Compare 1 ⁽²⁾	318	R/W	Yes	No
DVC2	Data Value Compare 2 ⁽²⁾	319	R/W	Yes	No
SPEFSCR	LSP/EFP APU status and control register	512	R/W	No	No
L1CFG0	L1 cache config register 0	515	Read-only	No	Yes
L1CFG1	L1 cache config register 1	516	Read-only	No	Yes
NPIDR	Nexus 3 Process ID register	517	R/W	No	Yes
DBCR3	Debug control register 3	561	R/W	Yes	Yes
DBCNT	Debug Counter register	562	R/W	Yes	Yes
DBCR4	Debug control register 4	563	R/W	Yes	Yes
DBCR5	Debug control register 5	564	R/W	Yes	Yes
IAC5	Instruction Address Compare 5	565	R/W	Yes	Yes
IAC6	Instruction Address Compare 6	566	R/W	Yes	Yes
IAC7	Instruction Address Compare 7	567	R/W	Yes	Yes
IAC8	Instruction Address Compare 8	568	R/W	Yes	Yes
MCSRR0	Machine Check Save/Restore Register 0	570	R/W	Yes	Yes
MCSRR1	Machine Check Save/Restore Register 1	571	R/W	Yes	Yes
MCSR	Machine Check Syndrome Register	572	Read/Clear ⁽³⁾	Yes	Yes
MCAR	Machine Check Address Register	573	R/W	Yes	Yes
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
DDAM	Debug Data Acquisition Messaging register	576	R/W	No	Yes
DAC3	Data Address Compare 3	592	R/W	Yes	Yes
DAC4	Data Address Compare 4	593	R/W	Yes	Yes
DBCR7	Debug control register 7	596	R/W	Yes	Yes
DBCR8	Debug control register 8	597	R/W	Yes	Yes
DDEAR	Debug Data Effective Address register	600	R/W	Yes	Yes
DVC1U ⁽⁴⁾	Data Value Compare 1 Upper	601	R/W	Yes	No
DVC2U ⁽⁴⁾	Data Value Compare 2 Upper	602	R/W	Yes	No
DBCR6	Debug control register 6	603	R/W	Yes	Yes
MAS0	MPU assist register 0	624	R/W	Yes	Yes
MAS1	MPU assist register 1	625	R/W	Yes	Yes
MAS2	MPU assist register 2	626	R/W	Yes	Yes
MAS3	MPU assist register 3	627	R/W	Yes	Yes

Table 24. Special Purpose Registers (SPRs) (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
EDBRAC0	External debug resource allocation control register 0	638	Read only	Yes	Yes
MPU0CFG	MPU0 configuration register	692	Read-only	Yes	Yes
L1FINV1	L1 cache flush and invalidate control register 0	959	R/W	Yes	Yes
DEVENT	Debug Event register	975	R/W	No	Yes
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
L1CSR0	L1 cache control and status register 0	1010	R/W	Yes	Yes
L1CSR1	L1 cache control and status register 1	1011	R/W	Yes	Yes
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes
MPU0CSR0	MPU0 configuration register	1014	R/W	Yes	Yes
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
L1FINV0	L1 cache flush and invalidate control register 0	1016	R/W	Yes	Yes
SVR	System Version Register	1023	Read-only	Yes	Yes

1. The Debug Status Register can be read using *mfscr RT,DBSR*. The Debug Status Register cannot be directly written to. Instead, bits in the Debug Status Register corresponding to '1' bits in GPR(RS) can be cleared using *mtscr DBSR,RS*.
2. Undefined on **POR** assertion, unchanged on **external PORESET** assertion.
3. The Machine Check Syndrome Register can be read using *mfscr RT,MCSR*. The Machine Check Syndrome Register cannot be directly written to. Instead, bits in the Machine Check Syndrome Register corresponding to '1' bits in GPR(RS) can be cleared using *mtscr MCSR,RS*.
4. The 64-bit DVC registers are accessed by using the DVC1,2 spr #s for the lower 32 bits, and the DVC1,2U spr #s for the upper 32 bits.

7.3 System modules

7.3.1 SIUL2 configuration

The System Integration Unit Lite 2 (SIUL2) controls:

- MCU pad configuration
- Ports
- General-purpose input and output (GPIO) signals
- External interrupts with trigger event configuration

For details please refer [Chapter 16: System Integration Unit Lite2 \(SIUL2\)](#).

7.3.2 Semaphores2 (SEMA42)

The peripheral platform shell includes a memory-mapped module (SEMA42) that provides

- Robust hardware support needed in multi-core systems for implementing semaphores
- A simple mechanism to achieve "lock/unlock" operations via a single write access

The hardware semaphore module is an architecture-neutral solution that provides sixteen hardware-enforced gates for SPC57EM80xx as well as other useful system functions related to the gating mechanisms. The capabilities provided by the hardware semaphore module are currently utilized in dual-core AUTOSAR® implementations and its use is expected to continue in future multi-core ports. The result is a second-generation module implementation known as Semaphores2 or SEMA42.

For details refer to [Chapter 23: Semaphores2 \(SEMA42\)](#).

7.3.3 Crossbar switch configuration

This device contains two crossbar switch modules. [Figure 9](#) illustrates that they are connected via a Cross-lake, which is comprised of intelligent bridging bus gaskets. These bus gaskets handle the traffic which crosses between the shells and do not require any interaction with the user. This dual-crossbar architecture allows all masters to access all slaves across the computational shell and peripheral shell.

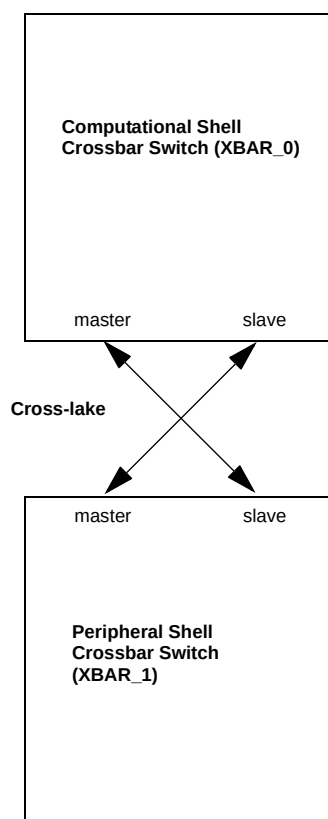


Figure 9. Crossbar switch diagram

The following sections explain how each crossbar switch is configured on this device.

7.3.3.1 Computational shell crossbar switch (XBAR_0) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

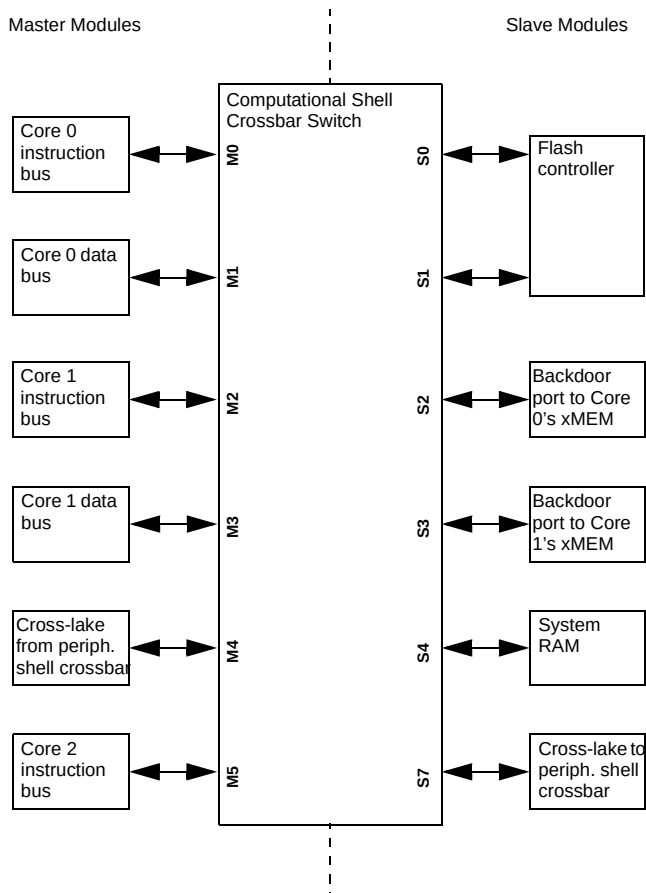


Figure 10. Computational shell's crossbar switch integration

Table 25. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	Chapter 17: Crossbar Switch (XBAR)
System memory map	—	Chapter 5: Memory Map
Clocking	—	Chapter 27: Clocking
Crossbar switch master	Cores	Section 7.2, Core modules
Crossbar switch slave	Flash memory controller	Section 3.3.1, Flash memory controller
Crossbar switch slave	Processor core local memory	Section 3.2.4, Processor core local SRAM
Crossbar switch slave	System RAM	Section 3.2.1, System SRAM

7.3.3.1.1 Computational shell crossbar switch master assignments

This device contains six masters connected to the computational shell's crossbar switch. The master assignment is shown in [Table 26](#).

Table 26. Computational shell crossbar switch master assignments

Master module	Physical master port number
Core 0 instruction bus	0
Core 0 data bus	1
Core 1 instruction bus	2
Core 1 data bus	3
Peripheral shell crossbar switch	4
Core 2 instruction bus	5

7.3.3.1.2 Computational shell crossbar switch slave assignments

This device contains six slaves connected to the computational shell's crossbar switch.

The slave assignment is shown in [Table 27](#).

Table 27. Computational shell crossbar switch slave assignments

Slave module	Physical slave port number
Flash memory controller port 0	0
Flash memory controller port 1	1
Backdoor port to Core 0's xMEM	2
Backdoor port to Core 1's xMEM	3
System RAM	4
Peripheral shell crossbar switch	7

Access to the flash memory via slave ports 0 and 1 is routed based on the master. The master-to-flash memory port routing have been assigned in a way to balance data traffic, with attention paid to facilitate core instruction fetches. The following table shows the flash memory port assignments.

Table 28. Flash memory port assignments

Flash memory slave port 0	Flash memory slave port 1
Core 0 instruction bus (M0)	Core 0 data bus (M1)
Core 1 data bus (M3)	Core 1 instruction bus (M2)
Peripheral shell crossbar switch (M4)	Core 2 instruction bus (M5)

7.3.3.1.3 PRS register reset values

The computational shell's XBAR_0 PRS_n registers reset to 0x0030_1425. Note that these registers are read-only.

7.3.3.2 Peripheral shell crossbar switch (XBAR_1) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

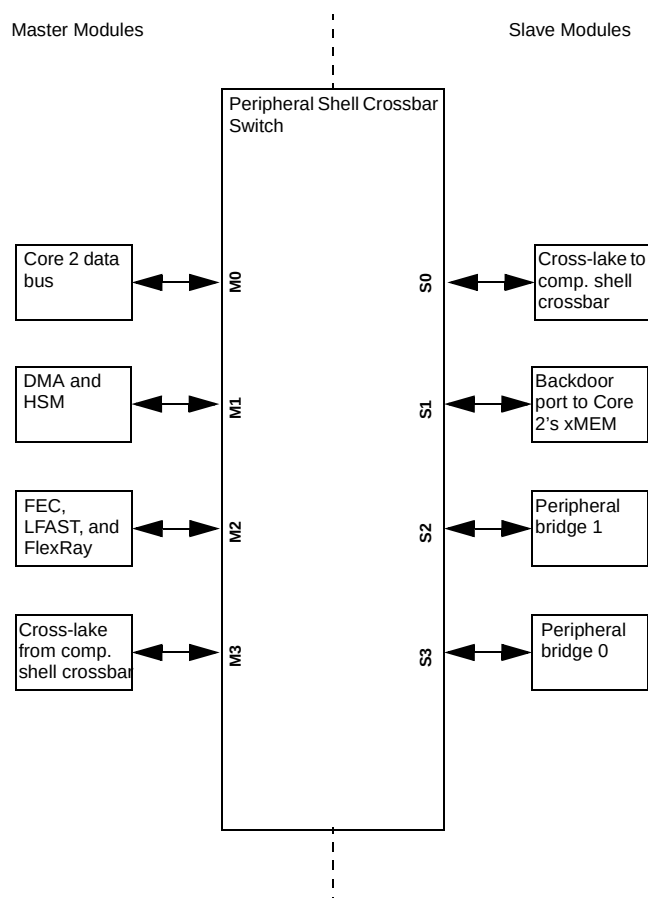


Figure 11. Crossbar switch integration

Table 29. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	Chapter 17: Crossbar Switch (XBAR)
System memory map	—	Chapter 5: Memory Map
Clocking	—	Chapter 27: Clocking
Crossbar switch master	Cores	Section 7.2, Core modules
Crossbar switch master	DMA and HSM	Section 7.3.8, DMA Controller configuration
Crossbar switch master	FEC, LFAST, and FlexRay	Section 7.8.1, Fast Ethernet Controller (FEC) configuration , Section 7.8.4, LFAST and SIPI , Section 7.8.6, FlexRay Configuration
Crossbar switch slave	Processor core local memory	Section 3.2.4, Processor core local SRAM
Crossbar switch slave	Peripheral bridge	Chapter 20: Peripheral Bridge

7.3.3.2.1 Peripheral shell crossbar switch master assignments

This device contains four masters connected to the peripheral shell's crossbar switch.

The master assignment is shown in [Table 30](#).

Table 30. Peripheral shell crossbar switch master assignments

Master module	Physical master port number
Core 2 data bus	0
DMA and Hardware Security Module (HSM)	1
FEC, LFAST, and FlexRay	2
Cross-lake (computational shell crossbar switch)	3

7.3.3.2.2 Peripheral shell crossbar switch slave assignments

This device contains four slaves connected to the peripheral shell's crossbar switch.

The slave assignment is shown in [Table 31](#).

Table 31. Peripheral shell crossbar switch slave assignments

Slave module	Physical slave port number
Computational shell crossbar switch	0
Backdoor port to Core 2's xMEM	1
Peripheral bridge 1	2
Peripheral bridge 0	3

7.3.3.2.3 PRS register reset values

The peripheral shell's XBAR_1 PRS_n registers reset to 0000_3102h.

7.3.4 System Memory Protection Unit (SMPU) configuration

This chip contains three instances of the System Memory Protection Unit (SMPU):

- SMPU_HSM for the Hardware Security Module
- SMPU_0 for the Computational Shell
- SMPU_1 for the Peripheral Shell

The following sections explain how SMPU_0 and SMPU_1 are configured on this chip. See the *SPC57EM80xx Microcontroller Security Reference Manual* for configuration details of SMPU_HSM.

7.3.4.1 SMPU0 configuration

This section summarizes how the module instance has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

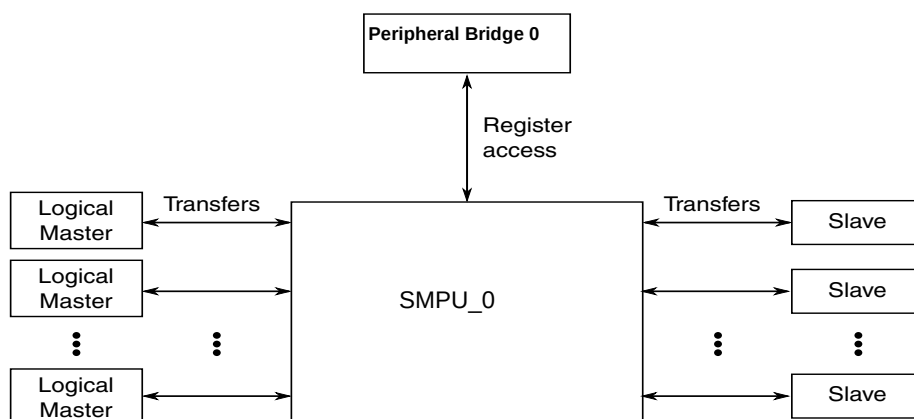


Figure 12. SMPU0 configuration

Table 32. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	Chapter 21: System Memory Protection Unit (SMPU)
System memory map	—	Chapter 5: Memory Map
Clocking	—	Chapter 27: Clocking
Power management	—	Chapter 10: Power management
Logical masters	—	Section 7.3.4.1.1, SMPU0 logical bus master assignments
Region descriptors	—	Section 7.3.4.1.2, Number of region descriptors supported by SMPU0
RGD_WORD2 formats	—	Section 7.3.4.1.3, RGD_WORD2 formats supported by SMPU0
Master Access Permissions	—	Section 7.3.4.1.4, Master Access permissions for SMPU0

7.3.4.1.1 SMPU0 logical bus master assignments

The logical bus master assignments for the SMPU0 are:

Table 33. SMPU0 logical bus master assignments

SMPU logical bus master number	Bus master
0	Core0
1	Core1
2	Core2
3	DMA
4	Ethernet
5	FlexRay

Table 33. SMPU0 logical bus master assignments (continued)

SMPU logical bus master number	Bus master
6	Reserved
7	SIPI/LFAST
8	Core0 Debug
9	Core1 Debug
10	Core2 Debug
11	Reserved
12	Reserved
13	Reserved
14	HSM
15	Reserved

7.3.4.1.2 Number of region descriptors supported by SMPU0

This instance of the SMPU module supports a maximum of twelve region descriptors.

7.3.4.1.3 RGD_WORD2 formats supported by SMPU0

For RGD_WORD2, this instance of the SMPU module supports only the FMT0 format. The RGD_WORD3[FMT] field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.

7.3.4.1.4 Master Access permissions for SMPU0

The only masters allowed to access the SMPU0 registers are HSM, Core 2, Core 1, and Core 0. It is also required that the core be in supervisor mode.

7.3.4.2 SMPU1 configuration

This section summarizes how the module instance has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

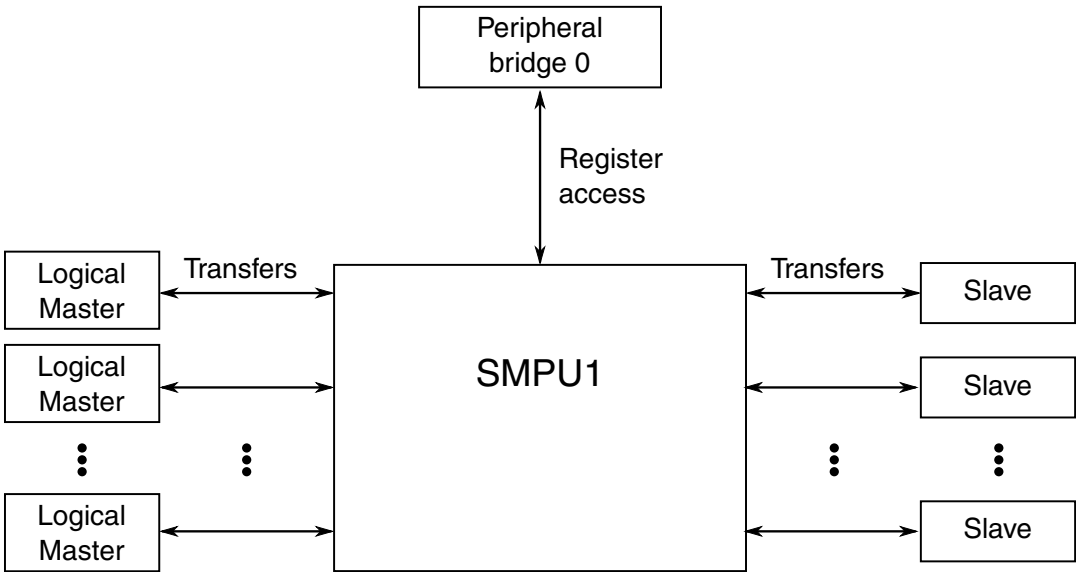


Figure 13. SMPU1 configuration

Table 34. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	Chapter 21: System Memory Protection Unit (SMPU)
System memory map	—	Chapter 5: Memory Map
Clocking	—	Chapter 27: Clocking
Power management	—	Chapter 10: Power management
Logical masters	—	Section 7.3.4.2.1, SMPU1 logical bus master assignments
Region descriptors	—	Section 7.3.4.2.2, Number of region descriptors supported by SMPU1
RGD_WORD2 formats	—	Section 7.3.4.2.3, RGD_WORD2 formats supported by SMPU1
Master Access Permissions	—	Section 7.3.4.2.4, Master Access permissions for SMPU1

7.3.4.2.1 SMPU1 logical bus master assignments

The logical bus master assignments for the SMPU1 are:

Table 35. SMPU1 logical bus master assignments

SMPU logical bus master number	Bus master
0	Core0
1	Core1
2	Core2

Table 35. SMPU1 logical bus master assignments (continued)

SMPU logical bus master number	Bus master
3	DMA
4	Ethernet
5	FlexRay
6	Reserved
7	SIPI/LFAST
8	Core0 Debug
9	Core1 Debug
10	Core2 Debug
11	Reserved
12	Reserved
13	Reserved
14	HSM
15	Reserved

7.3.4.2.2 Number of region descriptors supported by SMPU1

This instance of the SMPU module supports a maximum of eight region descriptors.

7.3.4.2.3 RGD_WORD2 formats supported by SMPU1

For RGD_WORD2, this instance of the SMPU module supports only the FMT0 format. The RGD_WORD3[FMT] field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.

7.3.4.2.4 Master Access permissions for SMPU1

The only masters allowed to access the SMPU1 registers are HSM, Core 2, Core 1, and Core 0. It is also required that the core be in supervisor mode.

7.3.5 Peripheral bridge configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

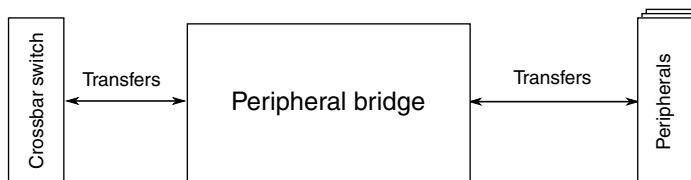


Figure 14. Peripheral bridge configuration

Table 36. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge	Chapter 20: Peripheral Bridge
System memory map	—	Chapter 5: Memory Map
Clocking	—	Chapter 27: Clocking
Crossbar switch	Crossbar switch	Section 7.3.3, Crossbar switch configuration

7.3.5.1 Number of peripheral bridges

This device contains two identical peripheral bridges.

7.3.5.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See [Chapter 5: Memory Map](#) for the memory slot assignment for each module.

7.3.5.3 MPR registers

Each of the two peripheral bridges supports a maximum of sixteen crossbar switch masters, each assigned to a MPROT n field in the MPRA-B registers. However, fewer are supported on this device. See [Section 7.3.3, Crossbar switch configuration](#) for details of the master port assignments for this device.

7.3.5.4 PACR/OPACR registers

The peripherals attached to the peripheral bridges each are assigned to a PACR n or OPACR n field within the PACRA-H/OPACRA-AF registers. See [Chapter 5: Memory Map](#) for details of the peripheral slot assignments for this device. Unused PACR n fields are reserved.

7.3.6 Platform Configuration Module (PCM)

The Platform Configuration Module contains three miscellaneous configuration registers for the device. Currently, the configuration registers are related to the operation of the FEC and

intelligent bus bridging gasket. The module is mapped to PBRIDGE on-platform slot 10 with a base address of 0xFC02_8000.

7.3.6.1 FEC Burst Optimization Master Control Register (FBOMCR)

Figure 15. FEC Burst Optimization Master Control Register (FBOMCR)

Offset:	00h				Access: Supervisor read/write											
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	ACCERR	WBEN	RBEN	FXSBE7	FXSBE6	FXSBE5	FXSBE4	FXSBE3	FXSBE2	FXSBE1	FXSBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16. FBOMCR field description

Field	Description
0–20	Reserved
21 ACCERR	Accumulate error This bit determines whether an error response for the first half of the write burst is accumulated to the second half of the write burst or discarded. In order to complete the burst, the FEC interface to the system bus responds by indicating that the first half of the burst completed without error before it actually writes the data so that it can fetch the second half of the write data from the FIFO. When actually written onto the system bus, the first half of the write burst can have an error. Because this half initially responded without an error to the FIFO, the error is discarded or accumulated with the error response for the second half of the burst. 0 Any error to the first half of the write burst is discarded. 1 Any actual error response to the first half of the write burst is accumulated in the second half's response. In other words, an error response to the first half will be seen in the response to the second half, even if the second half does not contain an error.
22 WBEN	Global write burst enable to XBAR slave port designated by FXSBE_n 0 Write bursting to all XBAR slave ports is disabled. 1 Write bursting is enabled to any XBAR slave port whose FXSBE_n bit is asserted.

Figure 16. FBOMCR field description (continued)

Field	Description
23 RBEN	Global read burst enable from XBAR slave port designated by FXSBE _n 0 Read bursting from all XBAR slave ports is disabled. 1 Read bursting is enabled from any XBAR slave port whose FXSBE _n bit is asserted.
24-31 FXSBE[7:0]	The FEC burst optimization master control register (FBOMCR) controls FEC burst optimization behavior on the system bus. FXSBE — FEC XBAR slave burst enable. FXSBE_n enables bursting by the FEC interface to the XBAR slave port controlled by that respective FXSBE_n bit. If FXSBE_n is asserted, then that XBAR slave port enabled by the bit can accept the bursts allowed by RBEN and WBEN. Otherwise, the FEC interface will not burst to the XBAR slave port controlled by that respective FXSBE_n bit. Read bursts from that XBAR slave port are enabled by RBEN. Write bursts to that XBAR slave port are enabled by WBEN. FXSBE_n assignments to XBAR slave ports: FXSBE0 = Flash FXSBE1 = reserved FXSBE2 = RAM, TCMA, TCMB, TCMC FXSBE3 = reserved FXSBE4 = reserved FXSBE5 = reserved FXSBE6 = reserved FXSBE7 = PBRIDGE0 and PBRIDGE1

7.3.6.2 IAHB Burst Enable 1 (IAHB_BE1) Register

Figure 17. IAHB Burst Enable 1 (IAHB_BE1) Register

Offset: 04h

Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0				0	0	0	0	0	PRE_DMA	BRE_DMA	BWE_DMA
W						RSV	RSV	RSV								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	PRE_S0	BRE_S0	BWE_S0	0	0	0	0	0	PRE_M3	BRE_M3	BWE_M3
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Table 37. IAHB_BE1 Register field description

Field	Description
0–4	Reserved
5–7 RSV	Reserved bits define the PRE, BRE and BWE of the bus gasket for the HSM.
8–12	Reserved

Table 37. IAHB_BE1 Register field description (continued)

Field	Description
13 PRE_DMA	Pending read enable (PRE) DMA This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_DMA	Burst read enable (BRE) DMA This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_DMA	Burst write enable (BWE) DMA This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20	Reserved
21 PRE_S0	Pending read enable (PRE) S0 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
22 BRE_S0	Burst read enable (BRE) S0 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_S0	Burst write enable (BWE) S0 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28	Reserved
29 PRE_M3	Pending read enable (PRE) M3 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_M3	Burst read enable (BRE) M3 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_M3	Burst write enable (BWE) M3 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

7.3.6.3 IAHB Burst Enable 2 (IAHB_BE2) Register

Figure 18. IAHB Burst Enable 2 (IAHB_BE2) Register

Offset: 08h

Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0				0	0	0	0	0			
W						RSV	RSV	RSV						PRE_SIPI	BRE_SIPI	BWE_SIPI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0				0	0	0	0	0			
W						PRE_Flexray	BRE_Flexray	BWE_Flexray						PRE_FEC	BRE_FEC	BWE_FEC
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Table 38. IAHB_BE2 field description

Field	Description
0–4	Reserved
5–7 RSV	Reserved, undefined.
8–12	Reserved
13 PRE_SIPI	Pending read enable (PRE) SIPI This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_SIPI	Burst read enable (BRE) SIPI This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_SIPI	Burst write enable (BWE) SIPI This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20	Reserved
21 PRE_Flexray	Pending read enable (PRE) Flexray This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.

Table 38. IAHB_BE2 field description (continued)

Field	Description
22 BRE_Flexray	Burst read enable (BRE) Flexray This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_Flexray	Burst write enable (BWE) Flexray This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28	Reserved
29 PRE_FEC	Pending read enable (PRE) FEC This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_FEC	Burst read enable (BRE) FEC This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_FEC	Burst write enable (BWE) FEC This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

7.3.7 Interrupt Controller (INTC) configuration

7.3.7.1 INTC configurable parameters

The INTC on this chip supports three processors—the safety core, the computational core, and the I/O processor (IOP).

7.3.7.2 INTC implemented registers

[Table 39](#) shows the registers implemented on the INTC on this chip.

Table 39. INTC implemented registers

Address offset	Register
000h	INTC Block Configuration Register (INTC_BCR)
010h	INTC Current Priority Register for Processor 0 (INTC_CPR0)
014h	INTC Current Priority Register for Processor 1 (INTC_CPR1)
018h	INTC Current Priority Register for Processor 2 (INTC_CPR2)
020h	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)
024h	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)

Table 39. INTC implemented registers (continued)

Address offset	Register
028h	INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)
030h	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)
034h	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)
038h	INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)
040h–05Ch	INTC Software Set/Clear Interrupt Register 0–3 (INTC_SSCIR0_3) – INTC Software Set/Clear Interrupt Register 28–31 (INTC_SSCIR28_31)
05Ch–85Ch	INTC Priority Select Register 0–1 (INTC_PSR0_1) – INTC Priority Select Register 1022–1023 (INTC_PSR1022_1023)

7.3.7.3 Interrupt sources

Table 40 defines the interrupt sources for the INTC on this chip. All IRQ numbers not specifically listed are not used.

Table 40. Interrupt sources

IRQ #	Offset	Source description	Source name
0	1000h	Software settable flag 0	SSCIR0_3[CLR0]
1	1004h	Software settable flag 1	SSCIR0_3[CLR1]
2	1008h	Software settable flag 2	SSCIR0_3[CLR2]
3	100Ch	Software settable flag 3	SSCIR0_3[CLR3]
4	1010h	Software settable flag 4	SSCIR4_7[CLR4]
5	1014h	Software settable flag 5	SSCIR4_7[CLR5]
6	1018h	Software settable flag 6	SSCIR4_7[CLR6]
7	101Ch	Software settable flag 7	SSCIR4_7[CLR7]
8	1020h	Software settable flag 8	SSCIR8_11[CLR8]
9	1024h	Software settable flag 9	SSCIR8_11[CLR9]
10	1028h	Software settable flag 10	SSCIR8_11[CLR10]
11	102Ch	Software settable flag 11	SSCIR8_11[CLR11]
12	1030h	Software settable flag 12	SSCIR12_15[CLR12]
13	1034h	Software settable flag 13	SSCIR12_15[CLR13]
14	1038h	Software settable flag 14	SSCIR12_15[CLR14]
15	103Ch	Software settable flag 15	SSCIR12_15[CLR15]
16	1040h	Software settable flag 16	SSCIR16_19[CLR16]
17	1044h	Software settable flag 17	SSCIR16_19[CLR17]
18	1048h	Software settable flag 18	SSCIR16_19[CLR18]
19	104Ch	Software settable flag 19	SSCIR16_19[CLR19]
20	1050h	Software settable flag 20	SSCIR20_23[CLR20]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
21	1054h	Software settable flag 21	SSCIR20_23[CLR21]
22	1058h	Software settable flag 22	SSCIR20_23[CLR22]
23	105Ch	Software settable flag 23	SSCIR20_23[CLR23]
24	1060h	Software settable flag 24	SSCIR24_27[CLR24]
25	1064h	Software settable flag 25	SSCIR24_27[CLR25]
26	1068h	Software settable flag 26	SSCIR24_27[CLR26]
27	106Ch	Software settable flag 27	SSCIR24_27[CLR27]
28	1070h	Software settable flag 28	SSCIR28_31[CLR28]
29	1074h	Software settable flag 29	SSCIR28_31[CLR29]
30	1078h	Software settable flag 30	SSCIR28_31[CLR30]
31	107Ch	Software settable flag 31	SSCIR28_31[CLR31]
32	1080h	Software watchdog timer 0 (SWT_0) Time-out	SWT_0_IR[TIF]
33	1084h	Software watchdog timer 1 (SWT_1) Time-out	SWT_1_IR[TIF]
34	1088h	Software watchdog timer 2 (SWT_2) Time-out	SWT_2_IR[TIF]
36	1090h	System Timer 0 Channel 0 match	STM_0_CIR0[CIF]
37	1094h	System Timer 0 Channel 1 match	STM_0_CIR1[CIF]
38	1098h	System Timer 0 Channel 2 match	STM_0_CIR2[CIF]
39	109Ch	System Timer 0 Channel 3 match	STM_0_CIR3[CIF]
40	10A0h	System Timer 1 Channel 0 match	STM_1_CIR0[CIF]
41	10A4h	System Timer 1 Channel 1 match	STM_1_CIR1[CIF]
42	10A8h	System Timer 1 Channel 2 match	STM_1_CIR2[CIF]
43	10ACh	System Timer 1 Channel 3 match	STM_1_CIR3[CIF]
44	10B0h	System Timer 2 Channel 0 match	STM_2_CIR0[CIF]
45	10B4h	System Timer 2 Channel 1 match	STM_2_CIR1[CIF]
46	10B8h	System Timer 2 Channel 2 match	STM_2_CIR2[CIF]
47	10BCh	System Timer 2 Channel 3 match	STM_2_CIR3[CIF]
52	10D0h	eDMA Combined Channel Error Flags 63-0	eDMA Channel Error Flags 63-0
53	10D4h	eDMA Channel interrupt 0	DMA_INTL[INT0]
54	10D8h	eDMA Channel interrupt 1	DMA_INTL[INT1]
55	10DCh	eDMA Channel interrupt 2	DMA_INTL[INT2]
56	10E0h	eDMA Channel interrupt 3	DMA_INTL[INT3]
57	10E4h	eDMA Channel interrupt 4	DMA_INTL[INT4]
58	10E8h	eDMA Channel interrupt 5	DMA_INTL[INT5]
59	10ECh	eDMA Channel interrupt 6	DMA_INTL[INT6]
60	10F0h	eDMA Channel interrupt 7	DMA_INTL[INT7]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
61	10F4h	eDMA Channel interrupt 8	DMA_INTL[INT8]
62	10F8h	eDMA Channel interrupt 9	DMA_INTL[INT9]
63	10FCh	eDMA Channel interrupt 10	DMA_INTL[INT10]
64	1100h	eDMA Channel interrupt 11	DMA_INTL[INT11]
65	1104h	eDMA Channel interrupt 12	DMA_INTL[INT12]
66	1108h	eDMA Channel interrupt 13	DMA_INTL[INT13]
67	110Ch	eDMA Channel interrupt 14	DMA_INTL[INT14]
68	1110h	eDMA Channel interrupt 15	DMA_INTL[INT15]
69	1114h	eDMA Channel interrupt 16	DMA_INTL[INT16]
70	1118h	eDMA Channel interrupt 17	DMA_INTL[INT17]
71	111Ch	eDMA Channel interrupt 18	DMA_INTL[INT18]
72	1120h	eDMA Channel interrupt 19	DMA_INTL[INT19]
73	1124h	eDMA Channel interrupt 20	DMA_INTL[INT20]
74	1128h	eDMA Channel interrupt 21	DMA_INTL[INT21]
75	112Ch	eDMA Channel interrupt 22	DMA_INTL[INT22]
76	1130h	eDMA Channel interrupt 23	DMA_INTL[INT23]
77	1134h	eDMA Channel interrupt 24	DMA_INTL[INT24]
78	1138h	eDMA Channel interrupt 25	DMA_INTL[INT25]
79	113Ch	eDMA Channel interrupt 26	DMA_INTL[INT26]
80	1140h	eDMA Channel interrupt 27	DMA_INTL[INT27]
81	1144h	eDMA Channel interrupt 28	DMA_INTL[INT28]
82	1148h	eDMA Channel interrupt 29	DMA_INTL[INT29]
83	114Ch	eDMA Channel interrupt 30	DMA_INTL[INT30]
84	1150h	eDMA Channel interrupt 31	DMA_INTL[INT31]
85	1154h	eDMA Channel interrupt 32	DMA_INTH[INT32]
86	1158h	eDMA Channel interrupt 33	DMA_INTH[INT33]
87	115Ch	eDMA Channel interrupt 34	DMA_INTH[INT34]
88	1160h	eDMA Channel interrupt 35	DMA_INTH[INT35]
89	1164h	eDMA Channel interrupt 36	DMA_INTH[INT36]
90	1168h	eDMA Channel interrupt 37	DMA_INTH[INT37]
91	116Ch	eDMA Channel interrupt 38	DMA_INTH[INT38]
92	1170h	eDMA Channel interrupt 39	DMA_INTH[INT39]
93	1174h	eDMA Channel interrupt 40	DMA_INTH[INT40]
94	1178h	eDMA Channel interrupt 41	DMA_INTH[INT41]
95	117Ch	eDMA Channel interrupt 42	DMA_INTH[INT42]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
96	1180h	eDMA Channel interrupt 43	DMA_INTH[INT43]
97	1184h	eDMA Channel interrupt 44	DMA_INTH[INT44]
98	1188h	eDMA Channel interrupt 45	DMA_INTH[INT45]
99	118Ch	eDMA Channel interrupt 46	DMA_INTH[INT46]
100	1190h	eDMA Channel interrupt 47	DMA_INTH[INT47]
101	1194h	eDMA Channel interrupt 48	DMA_INTH[INT48]
102	1198h	eDMA Channel interrupt 49	DMA_INTH[INT49]
103	119Ch	eDMA Channel interrupt 50	DMA_INTH[INT50]
104	11A0h	eDMA Channel interrupt 51	DMA_INTH[INT51]
105	11A4h	eDMA Channel interrupt 52	DMA_INTH[INT52]
106	11A8h	eDMA Channel interrupt 53	DMA_INTH[INT53]
107	11ACh	eDMA Channel interrupt 54	DMA_INTH[INT54]
108	11B0h	eDMA Channel interrupt 55	DMA_INTH[INT55]
109	11B4h	eDMA Channel interrupt 56	DMA_INTH[INT56]
110	11B8h	eDMA Channel interrupt 57	DMA_INTH[INT57]
111	11BCh	eDMA Channel interrupt 58	DMA_INTH[INT58]
112	11C0h	eDMA Channel interrupt 59	DMA_INTH[INT59]
113	11C4h	eDMA Channel interrupt 60	DMA_INTH[INT60]
114	11C8h	eDMA Channel interrupt 61	DMA_INTH[INT61]
115	11CCh	eDMA Channel interrupt 62	DMA_INTH[INT62]
116	11D0h	eDMA Channel interrupt 63	DMA_INTH[INT63]
185	12E4h	Flash controller Prog/Erase/Suspend IRQ_0	MCR[DONE]
189	12F4h	AMU DMA interrupt for AMU channel 0	DCTLCH0[DINT]
190	12F8h	AMU DMA interrupt for AMU channel 1	DCTLCH1[DINT]
191	12FCh	AMU Error interrupt for AMU channel 0	STCFG0[EINT]
192	1300h	AMU Error interrupt for AMU channel 1	STCFG1[EINT]
193	1304h	AMU Calculation interrupt for AMU channel 0	STCFG0[VINT] ⁽¹⁾ STCFG0[OINT] STCFG0[NINT]
194	1308h	AMU Calculation interrupt for AMU channel 1	STCFG1[VINT] STCFG1[OINT] STCFG1[NINT]
218	1368h	Ethernet Transmit frame interrupt	EIR[TXF]
219	136Ch	Ethernet Receive frame interrupt	EIR[RXF]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
220	1370h	Ethernet combined interrupt	EIR[HBERR] EIR[BABR] EIR[BABT] EIR[GRA] EIR[TXB] EIR[RXB] EIR[MII] EIR[EBERR] EIR[LC] EIR[RL] EIR[UN]
226	1388h	PIT0 Channel 0	PIT_0_TFLG0[TIF]
227	138Ch	PIT0 Channel 1	PIT_0_TFLG1[TIF]
228	1390h	PIT0 Channel 2	PIT_0_TFLG2[TIF]
229	1394h	PIT0 Channel 3	PIT_0_TFLG3[TIF]
230	1398h	PIT0 Channel 4	PIT_0_TFLG4[TIF]
231	139Ch	PIT0 Channel 5	PIT_0_TFLG5[TIF]
232	13A0h	PIT0 Channel 6	PIT_0_TFLG6[TIF]
233	13A4h	PIT0 Channel 7	PIT_0_TFLG7[TIF]
239	13BCh	PIT0 RTI Channel	PIT_0_RTI_TFLG[TIF]
240	13C0h	PIT1 Channel 0	PIT_1_TFLG0[TIF]
241	13C4h	PIT1 Channel 1	PIT_1_TFLG1[TIF]
242	13C8h	Crystal oscillator (XOSC) clock interrupt	XOSC_CTL[I_OSC]
243	13CCh	SIUL2 Combined External Interrupt 0 (= OR of REQ5, REQ4, REQ2, REQ1, REQ0)	SIUL2_DISR0[EIF5] SIUL2_DISR0[EIF4] SIUL2_DISR0[EIF2] SIUL2_DISR0[EIF1] SIUL2_DISR0[EIF0]
244	13D0h	SIUL2 Combined External Interrupt 1 (= OR of REQ10, REQ9, REQ8)	SIUL2_DISR0[EIF10] SIUL2_DISR0[EIF9] SIUL2_DISR0[EIF8]
251	13ECh	MC_ME Safe Mode interrupt	ME_IS[I_SAFE]
252	13F0h	MC_ME Mode Transition Interrupt	ME_IS[I_MTC]
253	13F4h	MC_ME Invalid Mode interrupt	ME_IS[I_IMODE]
254	13F8h	MC_ME Invalid Mode Config	ME_IS[I_ICONF]
255	13FCh	MC_RGM 0	MC_RGM Functional and destructive reset alternate event interrupt
259	140Ch	DSPI_0 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow	DSPI_0_SR[TFUF] DSPI_0_SR[RFOF]
260	1410h	DSPI_0 transmit FIFO end-of-queue flag	DSPI_0_SR[EOQF]
261	1414h	DSPI_0 Tx FIFO fill request flag	DSPI_0_SR[TFFF]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
262	1418h	DSPI_0 transfer complete flag	DSPI_0_SR[TCF]
263	141Ch	DSPI_0 Rx FIFO drain request flag	DSPI_0_SR[RDFD]
266	1428h	DSPI_0 combined DSI and SPI Parity Error Flags	DSPI_0_SR[SPEF]
268	1430h	DSPI_1 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow	DSPI_1_SR[TFUF] DSPI_1_SR[RFOF]
269	1434h	DSPI_1 transmit FIFO end-of-queue flag	DSPI_1_SR[EOQF]
270	1438h	DSPI_1 Tx FIFO fill request flag	DSPI_1_SR[TFFF]
271	143Ch	DSPI_1 transfer complete flag	DSPI_1_SR[TCF]
272	1440h	DSPI_1 Rx FIFO drain request flag	DSPI_1_SR[RDFD]
275	144Ch	DSPI_1 combined DSI and SPI Parity Error Flags	DSPI_1_SR[SPEF]
277	1454h	DSPI_2 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow	DSPI_2_SR[TFUF] DSPI_2_SR[RFOF]
278	1458h	DSPI_2 transmit FIFO end-of-queue flag	DSPI_2_SR[EOQF]
279	145Ch	DSPI_2 Tx FIFO fill request flag	DSPI_2_SR[TFFF]
280	1460h	DSPI_2 transfer complete flag	DSPI_2_SR[TCF]
281	1464h	DSPI_2 Rx FIFO drain request flag	DSPI_2_SR[RDFD]
284	1470h	DSPI_2 combined DSI and SPI Parity Error Flags	DSPI_2_SR[SPEF]
286	1478h	DSPI_3 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow	DSPI_3_SR[TFUF] DSPI_3_SR[RFOF]
287	147Ch	DSPI_3 transmit FIFO end-of-queue flag	DSPI_3_SR[EOQF]
288	1480h	DSPI_3 Tx FIFO fill request flag	DSPI_3_SR[TFFF]
289	1484h	DSPI_3 transfer complete flag	DSPI_3_SR[TCF]
290	1488h	DSPI_3 Rx FIFO drain request flag	DSPI_3_SR[RDFD]
293	1494h	DSPI_3 combined DSI and SPI Parity Error Flags	DSPI_3_SR[SPEF]
295	149Ch	DSPI_4 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow	DSPI_4_SR[TFUF] DSPI_4_SR[RFOF]
296	14A0h	DSPI_4 transmit FIFO end-of-queue flag	DSPI_4_SR[EOQF]
297	14A4h	DSPI_4 Tx FIFO fill request flag	DSPI_4_SR[TFFF]
298	14A8h	DSPI_4 transfer complete flag	DSPI_4_SR[TCF]
299	14ACh	DSPI_4 Rx FIFO drain request flag	DSPI_4_SR[RDFD]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
300	14B0h	DSPI_4 SPI Frame Transfer Complete Flag	DSPI_4_SR[SPITCF]
301	14B4h	DSPI_4 DSI Frame Transfer Complete Flag	DSPI_4_SR[DSITCF]
302	14B8h	DSPI_4 combined DSI and SPI Parity Error Flags	DSPI_4_SR[SPEF] DSPI_4_SR[DPEF]
303	14BCh	DSPI_4 DSI Data Received with Active Bits	DSPI_4_SR[DDIF]
304	14C0h	DSPI_5 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow	DSPI_5_SR[TFUF] DSPI_5_SR[RFOF]
305	14C4h	DSPI_5 transmit FIFO end-of-queue flag	DSPI_5_SR[EOQF]
306	14C8h	DSPI_5 Tx FIFO fill request flag	DSPI_5_SR[TFFF]
307	14CCh	DSPI_5 transfer complete flag	DSPI_5_SR[TCF]
308	14D0h	DSPI_5 Rx FIFO drain request flag	DSPI_5_SR[RFDF]
309	14D4h	DSPI_5 SPI Frame Transfer Complete Flag	DSPI_5_SR[SPITCF]
310	14D8h	DSPI_5 DSI Frame Transfer Complete Flag	DSPI_5_SR[DSITCF]
311	14DCh	DSPI_5 combined DSI and SPI Parity Error Flags	DSPI_5_SR[SPEF] DSPI_5_SR[DPEF]
312	14E0h	DSPI_5 DSI Data Received with Active Bits	DSPI_5_SR[DDIF]
367	15BCh	DSPI_12 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow	DSPI_12_SR[TFUF] DSPI_12_SR[RFOF]
368	15C0h	DSPI_12 transmit FIFO end-of-queue flag	DSPI_12_SR[EOQF]
369	15C4h	DSPI_12 Tx FIFO fill request flag	DSPI_12_SR[TFFF]
370	15C8h	DSPI_12 transfer complete flag	DSPI_12_SR[TCF]
371	15CCh	DSPI_12 Rx FIFO drain request flag	DSPI_12_SR[RFDF]
374	15D8h	DSPI_12 combined DSI and SPI Parity Error Flags	DSPI_12_SR[SPEF]
376	15E0h	LINFlexD_0 Reception combined interrupt: – Header Received – Data Reception complete – Data Buffer Full – Wake-up – LIN State (Sync Del, Sync Field, Identifier Field or Checksum Field)	LINFlexD_0_LINIER[HRIE] LINFlexD_0_LINIER[DRIE] LINFlexD_0_LINIER[DBFIE] LINFlexD_0_LINIER[WUIE] LINFlexD_0_LINIER[LSIE]
377	15E4h	LINFlexD_0 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_0_LINIER[HRIE] LINFlexD_0_LINIER[DTIE] LINFlexD_0_LINIER[DBEIE]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
378	15E8h	LINFlexD_0 Error combined interrupt: – Stuck at zero – Output Compare – Bit Error – Checksum Error – Header Error – Frame Error – Buffer Overrun – UART Timeout Error	LINFlexD_0_LINIER[SZIE] LINFlexD_0_LINIER[OCIE] LINFlexD_0_LINIER[BEIE] LINFlexD_0_LINIER[CEIE] LINFlexD_0_LINIER[HEIE] LINFlexD_0_LINIER[FEIE] LINFlexD_0_LINIER[BOIE] LINFlexD_0_LINIER[TOIE]
380	15F0h	LINFlexD_1 Reception combined interrupt: – Header Received – Data Reception complete – Data Buffer Full – Wake-up – LIN State (Sync Del, Sync Field, Identifier Field or Checksum Field)	LINFlexD_1_LINIER[HRIE] LINFlexD_1_LINIER[DRIE] LINFlexD_1_LINIER[DBFIE] LINFlexD_1_LINIER[WUIE] LINFlexD_1_LINIER[LSIE]
381	15F4h	LINFlexD_1 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_1_LINIER[HRIE] LINFlexD_1_LINIER[DTIE] LINFlexD_1_LINIER[DBEIE]
382	15F8h	LINFlexD_1 Error combined interrupt: – Stuck at zero – Output Compare – Bit Error – Checksum Error – Header Error – Frame Error – Buffer Overrun – UART Timeout Error	LINFlexD_1_LINIER[SZIE] LINFlexD_1_LINIER[OCIE] LINFlexD_1_LINIER[BEIE] LINFlexD_1_LINIER[CEIE] LINFlexD_1_LINIER[HEIE] LINFlexD_1_LINIER[FEIE] LINFlexD_1_LINIER[BOIE] LINFlexD_1_LINIER[TOIE]
384	1600h	LINFlexD_2 Reception combined interrupt: – Header Received – Data Reception complete – Data Buffer Full – Wake-up – LIN State (Sync Del, Sync Field, Identifier Field or Checksum Field)	LINFlexD_2_LINIER[HRIE] LINFlexD_2_LINIER[DRIE] LINFlexD_2_LINIER[DBFIE] LINFlexD_2_LINIER[WUIE] LINFlexD_2_LINIER[LSIE]
385	1604h	LINFlexD_2 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_2_LINIER[HRIE] LINFlexD_2_LINIER[DTIE] LINFlexD_2_LINIER[DBEIE]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
386	1608h	LINFlexD_2 Error combined interrupt: – Stuck at zero – Output Compare – Bit Error – Checksum Error – Header Error – Frame Error – Buffer Overrun – UART Timeout Error	LINFlexD_2_LINIER[SZIE] LINFlexD_2_LINIER[OCIE] LINFlexD_2_LINIER[BEIE] LINFlexD_2_LINIER[CEIE] LINFlexD_2_LINIER[HEIE] LINFlexD_2_LINIER[FEIE] LINFlexD_2_LINIER[BOIE] LINFlexD_2_LINIER[TOIE]
432	16C0h	LINFlexD_14 Reception combined interrupt: – Header Received – Data Reception complete – Data Buffer Full – Wake-up – LIN State (Sync Del, Sync Field, Identifier Field or Checksum Field)	LINFlexD_14_LINIER[HRIE] LINFlexD_14_LINIER[DRIE] LINFlexD_14_LINIER[DBFIE] LINFlexD_14_LINIER[WUIE] LINFlexD_14_LINIER[LSIE]
433	16C4h	LINFlexD_14 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_14_LINIER[HRIE] LINFlexD_14_LINIER[DTIE] LINFlexD_14_LINIER[DBEIE]
434	16C8h	LINFlexD_14 Error combined interrupt: – Stuck at zero – Output Compare – Bit Error – Checksum Error – Header Error – Frame Error – Buffer Overrun – UART Timeout Error	LINFlexD_14_LINIER[SZIE] LINFlexD_14_LINIER[OCIE] LINFlexD_14_LINIER[BEIE] LINFlexD_14_LINIER[CEIE] LINFlexD_14_LINIER[HEIE] LINFlexD_14_LINIER[FEIE] LINFlexD_14_LINIER[BOIE] LINFlexD_14_LINIER[TOIE]
436	16D0h	LINFlexD_15 Reception combined interrupt: – Header Received – Data Reception complete – Data Buffer Full – Wake-up – LIN State (Sync Del, Sync Field, Identifier Field or Checksum Field)	LINFlexD_15_LINIER[HRIE] LINFlexD_15_LINIER[DRIE] LINFlexD_15_LINIER[DBFIE] LINFlexD_15_LINIER[WUIE] LINFlexD_15_LINIER[LSIE]
437	16D4h	LINFlexD_15 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_15_LINIER[HRIE] LINFlexD_15_LINIER[DTIE] LINFlexD_15_LINIER[DBEIE]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
438	16D8h	LINFlexD_15 Error combined interrupt: – Stuck at zero – Output Compare – Bit Error – Checksum Error – Header Error – Frame Error – Buffer Overrun – UART Timeout Error	LINFlexD_15_LINIER[SZIE] LINFlexD_15_LINIER[OCIE] LINFlexD_15_LINIER[BEIE] LINFlexD_15_LINIER[CEIE] LINFlexD_15_LINIER[HEIE] LINFlexD_15_LINIER[FEIE] LINFlexD_15_LINIER[BOIE] LINFlexD_15_LINIER[TOIE]
440	16E0h	I2C Combined interrupt: – Byte transfer complete – Addressed as slave – Arbitration Lost – No Acknowledge from Slave – I2C Bus going idle	I2C_0_IBSR[IBIF] I2C_0_IBSR[IAAS] I2C_0_IBSR[IBAL]
453	1714h	FlexRay LRAM Non-Corrected errors combined interrupt: – LRAM Non-Corrected Error Overflow – LRAM Non-Corrected Error	FR_0_EEIFER[LRNE_OF] FR_0_EEIFER[LRNE_IF]
454	1718h	FlexRay LRAM Corrected errors combined interrupt: – LRAM Corrected Error Overflow – LRAM Corrected Error	FR_0_EEIFER[LRCE_OF] FR_0_EEIFER[LRCE_IF]
455	171Ch	FlexRay Receive FIFO Channel A Almost Full Interrupt	FR_0_GIFER[FAFAIF]
456	1720h	FlexRay Receive FIFO Channel B Almost Full Interrupt	FR_0_GIFER[FAFBIF]
457	1724h	FlexRay Wakeup Interrupt	FR_0_GIFER[WUPIF]
458	1728h	FlexRay Protocol global interrupt	FR_0_GIFER[PRIF]
459	172Ch	FlexRay Controller Host Interface global interrupt	FR_0_GIFER[CHIF]
460	1730h	FlexRay Transmit Message Buffer global Interrupt	FR_0_GIFER[TBIF]
461	1734h	FlexRay Receive Message Buffer global Interrupt	FR_0_GIFER[RBIF]
462	1738h	FlexRay Module global Interrupt	FR_0_GIFER[MIF]
477	1774h	Power Management Controller Supply Gauge Low Voltage Detect and High Voltage Detect Interrupts	GR_S[VD15] GR_S[VD14] GR_S[VD13] GR_S[VD12] GR_S[VD10] GR_S[VD9] GR_S[VD7] GR_S[VD4] GR_S[VD3]
478	1778h	Power Management Controller Temperature Sensor interrupts	EPR_TD[TEMP_3] EPR_TD[TEMP_2] EPR_TD[TEMP_0]
480	1780h	Loss-of-lock interrupt request for PLL0	PLL0SR[LOLF]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
482	1788h	Loss-of-lock interrupt request for PLL1	PLL1SR[LOLF]
484	1790h	PLL0 External Power Down Cycle Complete indication interrupt	PLL0SR[EXTPDF]
485	1794h	PLL1 External Power Down Cycle Complete indication interrupt	PLL1SR[EXTPDF]
488	17A0h	FCCU Alarm interrupt	FCCU_IRQ_STAT[ALRM_STAT]
489	17A4h	FCCU_1	irq_misc_b
490	17A8h	FCCU_2	irq_rccs_b[0]
491	17ACh	FCCU_3	irq_rccs_b[1]
494	17B8h	STCU2 online self test (LBIST)	stcu_irq_lbist
495	17BCh	STCU2 online self test (MBIST)	stcu_irq_mbist
496	17C0h	Hardware Security Module 0	HSM2HTIE[0]
497	17C4h	Hardware Security Module 1	HSM2HTIE[1]
498	17C8h	Hardware Security Module 2	HSM2HTIE[2]
499	17CCh	Hardware Security Module 3	HSM2HTIE[3]
500	17D0h	Hardware Security Module 4	HSM2HTIE[4]
501	17D4h	Hardware Security Module 5	HSM2HTIE[5]
502	17D8h	Hardware Security Module 6	HSM2HTIE[6]
503	17DCh	Hardware Security Module 7	HSM2HTIE[7]
504	17E0h	Hardware Security Module 8	HSM2HTIE[8]
505	17E4h	Hardware Security Module 9	HSM2HTIE[9]
506	17E8h	Hardware Security Module 10	HSM2HTIE[10]
507	17ECh	Hardware Security Module 11	HSM2HTIE[11]
508	17F0h	Hardware Security Module 12	HSM2HTIE[12]
509	17F4h	Hardware Security Module 13	HSM2HTIE[13]
510	17F8h	Hardware Security Module 14	HSM2HTIE[14]
511	17FCh	Hardware Security Module 15	HSM2HTIE[15]
512	1800h	Hardware Security Module 16	HSM2HTIE[16]
513	1804h	Hardware Security Module 17	HSM2HTIE[17]
514	1808h	Hardware Security Module 18	HSM2HTIE[18]
515	180Ch	Hardware Security Module 19	HSM2HTIE[19]
516	1810h	Hardware Security Module 20	HSM2HTIE[20]
517	1814h	Hardware Security Module 21	HSM2HTIE[21]
518	1818h	Hardware Security Module 22	HSM2HTIE[22]
519	181Ch	Hardware Security Module 23	HSM2HTIE[23]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
520	1820h	Hardware Security Module 24	HSM2HTIE[24]
521	1824h	Hardware Security Module 25	HSM2HTIE[25]
522	1828h	Hardware Security Module 26	HSM2HTIE[26]
523	182Ch	Hardware Security Module 27	HSM2HTIE[27]
524	1830h	Hardware Security Module 28	HSM2HTIE[28]
525	1834h	Hardware Security Module 29	HSM2HTIE[29]
526	1838h	Hardware Security Module 30	HSM2HTIE[30]
527	183Ch	Hardware Security Module 31	HSM2HTIE[31]
528	1840h	SAR ADC0 combined interrupt: – NEOC (end of normal conversion of each channel) – NECH (end of normal chain) – JEOC (end of injected conversion of each channel) – JECH (end of injected chain)	SARADC_0_ICIPR0[EOC_CH[7]] SARADC_0_ICIPR0[EOC_CH[6]] SARADC_0_ICIPR0[EOC_CH[5]] SARADC_0_ICIPR0[EOC_CH[4]]
529	1844h	SAR ADC1 combined interrupt: – NEOC (end of normal conversion of each channel) – NECH (end of normal chain) – JEOC (end of injected conversion of each channel) – JECH (end of injected chain)	SARADC_1_ICIPR0[EOC_CH[15]] SARADC_1_ICIPR0[EOC_CH[14]] SARADC_1_ICIPR0[EOC_CH[13]] SARADC_1_ICIPR0[EOC_CH[12]] SARADC_1_ICIPR0[EOC_CH[11]] SARADC_1_ICIPR0[EOC_CH[10]] SARADC_1_ICIPR0[EOC_CH[9]] SARADC_1_ICIPR0[EOC_CH[8]]
530	1848h	SAR ADC2 combined interrupt: – NEOC (end of normal conversion of each channel) – NECH (end of normal chain) – JEOC (end of injected conversion of each channel) – JECH (end of injected chain)	SARADC_2_ICIPR0[EOC_CH[23]] SARADC_2_ICIPR0[EOC_CH[22]] SARADC_2_ICIPR0[EOC_CH[19]] SARADC_2_ICIPR0[EOC_CH[18]] SARADC_2_ICIPR0[EOC_CH[17]] SARADC_2_ICIPR0[EOC_CH[16]]
531	184Ch	SAR ADC3 combined interrupt: – NEOC (end of normal conversion of each channel) – NECH (end of normal chain) – JEOC (end of injected conversion of each channel) – JECH (end of injected chain)	SARADC_3_ICIPR0[EOC_CH[31]] SARADC_3_ICIPR0[EOC_CH[30]] SARADC_3_ICIPR0[EOC_CH[29]] SARADC_3_ICIPR0[EOC_CH[28]] SARADC_3_ICIPR0[EOC_CH[27]] SARADC_3_ICIPR0[EOC_CH[26]]
532	1850h	SAR ADC4 combined interrupt: – NEOC (end of normal conversion of each channel) – NECH (end of normal chain) – JEOC (end of injected conversion of each channel) – JECH (end of injected chain)	SARADC_4_ICIPR0[EOC_CH[35]] SARADC_4_ICIPR0[EOC_CH[34]] SARADC_4_ICIPR0[EOC_CH[33]] SARADC_4_ICIPR0[EOC_CH[32]]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
534	1858h	SAR_ADC6 combined interrupt: – NEOC (end of normal conversion of each channel) – NECH (end of normal chain) – JEOC (end of injected conversion of each channel) – JECH (end of injected chain)	SARADC_6_ICIPR0[EOC_CH[51]] SARADC_6_ICIPR0[EOC_CH[50]] SARADC_6_ICIPR0[EOC_CH[49]] SARADC_6_ICIPR0[EOC_CH[48]] SARADC_6_ICIPR0[EOC_CH[43]] SARADC_6_ICIPR0[EOC_CH[42]] SARADC_6_ICIPR0[EOC_CH[41]] SARADC_6_ICIPR0[EOC_CH[40]]
535	185Ch	SAR_ADC7 combined interrupt: – NEOC (end of normal conversion of each channel) – NECH (end of normal chain) – JEOC (end of injected conversion of each channel) – JECH (end of injected chain)	SARADC_7_ICIPR0[EOC_CH[59]] SARADC_7_ICIPR0[EOC_CH[58]] SARADC_7_ICIPR0[EOC_CH[57]] SARADC_7_ICIPR0[EOC_CH[56]]
543	187Ch	SAR_ADCB combined interrupt: – NEOC (end of normal conversion of each channel) – NECH (end of normal chain) – JEOC (end of injected conversion of each channel) – JECH (end of injected chain)	SARADC_B_ICIPR0[EOC_CH[31]] SARADC_B_ICIPR0[EOC_CH[30]] ... SARADC_B_ICIPR0[EOC_CH[1]] SARADC_B_ICIPR0[EOC_CH[0]] SARADC_B_ICIPR1[EOC_CH[61]] ... SARADC_B_ICIPR1[EOC_CH[56]] SARADC_B_ICIPR1[EOC_CH[53]] ... SARADC_B_ICIPR1[EOC_CH[0]]
544	1880h	SD ADC0 combined interrupt: – Data FIFO empty – Data FIFO full – Data FIFO overrun	SDADC_0_SFR[DFEF] SDADC_0_SFR[DFORF] SDADC_0_SFR[DFFF]
545	1884h	SD ADC1 combined interrupt: – Data FIFO empty – Data FIFO full – Data FIFO overrun	SDADC_1_SFR[DFEF] SDADC_1_SFR[DFORF] SDADC_1_SFR[DFFF]
546	1888h	SD ADC2 combined interrupt: – Data FIFO empty – Data FIFO full – Data FIFO overrun	SDADC_2_SFR[DFEF] SDADC_2_SFR[DFORF] SDADC_2_SFR[DFFF]
547	188Ch	SD ADC3 combined interrupt: – Data FIFO empty – Data FIFO full – Data FIFO overrun	SDADC_3_SFR[DFEF] SDADC_3_SFR[DFORF] SDADC_3_SFR[DFFF]
548	1890h	SD ADC4 combined interrupt: – Data FIFO empty – Data FIFO full – Data FIFO overrun	SDADC_4_SFR[DFEF] SDADC_4_SFR[DFORF] SDADC_4_SFR[DFFF]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
549	1894h	SD ADC5 combined interrupt: – Data FIFO empty – Data FIFO full – Data FIFO overrun	SDADC_5_SFR[DFEF] SDADC_5_SFR[DFORF] SDADC_5_SFR[DFFF]
558	18B8h	SENT_0 Fast messages combined interrupt: OR of all enabled Fast message Ready flags	SENT_0_FMSG_RDY[0] SENT_0_FMSG_RDY[1] SENT_0_FMSG_RDY[2] SENT_0_FMSG_RDY[3] SENT_0_FMSG_RDY[4]
559	18BCh	SENT_0 Slow messages combined interrupt: OR of all enabled Slow message Ready flags	SENT_0_SMSG_RDY[0] SENT_0_SMSG_RDY[1] SENT_0_SMSG_RDY[2] SENT_0_SMSG_RDY[3] SENT_0_SMSG_RDY[4]
560	18C0h	SENT_0 Error combined interrupt: OR of following events: – Fast message DMA underflow – Slow message DMA underflow – Channels 0 to 4: - Successive calibration check resynchronized error - Calibration variation 20 – 25% - Slow message overflow - Fast message overflow - Number of negative edges error - Checksum error in fast message - Checksum error in slow message - Nibble data value error - Calibration pulse length error - Calibration pulse length ratio to message length error	SENT_0_GBL_STATUS[FMDU] SENT_0_GBL_STATUS[SMDU] SENT_0_CHn_STATUS[4:7] SENT_0_CHn_STATUS[9:15]
561	18C4h	SENT_1 Fast messages combined interrupt: OR of all enabled Fast message Ready flags	SENT_1_FMSG_RDY[0] SENT_1_FMSG_RDY[1] SENT_1_FMSG_RDY[2] SENT_1_FMSG_RDY[3] SENT_1_FMSG_RDY[4]
562	18C8h	SENT_1 Slow messages combined interrupt: OR of all enabled Slow message Ready flags	SENT_1_SMSG_RDY[0] SENT_1_SMSG_RDY[1] SENT_1_SMSG_RDY[2] SENT_1_SMSG_RDY[3] SENT_1_SMSG_RDY[4]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
563	18CCh	SENT_1 Error combined interrupt: OR of following events: – Fast message DMA underflow – Slow message DMA underflow – Channels 0 to 4: - Successive calibration check resynchronized error - Calibration variation 20 – 25% - Slow message overflow - Fast message overflow - Number of negative edges error - Checksum error in fast message - Checksum error in slow message - Nibble data value error - Calibration pulse length error - Calibration pulse length ratio to message length error	SENT_1_GBL_STATUS[FMDU] SENT_1_GBL_STATUS[SMDU] SENT_1_CHn_STATUS[4:7] SENT_1_CHn_STATUS[9:15]
564	18D0h	SENT_0 Channel 0 Fast message interrupt	SENT_0_FMSG_RDY[0]
565	18D4h	SENT_0 Channel 0 Slow message interrupt	SENT_0_SMSG_RDY[0]
566	18D8h	SENT_0 Channel 0 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_0_CH0_STATUS[4:7] SENT_0_CH0_STATUS[9:15]
567	18DCh	SENT_0 Channel 1 Fast message interrupt	SENT_0_FMSG_RDY[1]
568	18E0h	SENT_0 Channel 1 Slow message interrupt	SENT_0_SMSG_RDY[1]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
569	18E4h	SENT_0 Channel 1 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_0_CH1_STATUS[4:7] SENT_0_CH1_STATUS[9:15]
570	18E8h	SENT_0 Channel 2 Fast message interrupt	SENT_0_FMSG_RDY[2]
571	18ECh	SENT_0 Channel 2 Slow message interrupt	SENT_0_SMSG_RDY[2]
572	18F0h	SENT_0 Channel 2 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_2_CH2_STATUS[4:7] SENT_2_CH2_STATUS[9:15]
573	18F4h	SENT_0 Channel 3 Fast message interrupt	SENT_0_FMSG_RDY[3]
574	18F8h	SENT_0 Channel 3 Slow message interrupt	SENT_0_SMSG_RDY[3]
575	18FCh	SENT_0 Channel 3 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_0_CH3_STATUS[4:7] SENT_0_CH3_STATUS[9:15]
576	1900h	SENT_0 Channel 4 Fast message interrupt	SENT_0_FMSG_RDY[4]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
577	1904h	SENT_0 Channel 4 Slow message interrupt	SENT_0_SMSG_RDY[4]
578	1908h	SENT_0 Channel 4 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_0_CH4_STATUS[4:7] SENT_0_CH4_STATUS[9:15]
579	190Ch	SENT_1 Channel 0 Fast message interrupt	SENT_1_FMSG_RDY[0]
580	1910h	SENT_1 Channel 0 Slow message interrupt	SENT_1_SMSG_RDY[0]
581	1914h	SENT_1 Channel 0 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_1_CH0_STATUS[4:7] SENT_1_CH0_STATUS[9:15]
582	1918h	SENT_1 Channel 1 Fast message interrupt	SENT_1_FMSG_RDY[1]
583	191Ch	SENT_1 Channel 1 Slow message interrupt	SENT_1_SMSG_RDY[1]
584	1920h	SENT_1 Channel 1 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_1_CH1_STATUS[4:7] SENT_1_CH1_STATUS[9:15]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
585	1924h	SENT_1 Channel 2 Fast message interrupt	SENT_1_FMSG_RDY[2]
586	1928h	SENT_1 Channel 2 Slow message interrupt	SENT_1_SMSG_RDY[2]
587	192Ch	SENT_1 Channel 2 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_1_CH2_STATUS[4:7] SENT_1_CH2_STATUS[9:15]
588	1930h	SENT_1 Channel 3 Fast message interrupt	SENT_1_FMSG_RDY[3]
589	1934h	SENT_1 Channel 3 Slow message interrupt	SENT_1_SMSG_RDY[3]
590	1938h	SENT_1 Channel 3 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_1_CH3_STATUS[4:7] SENT_1_CH3_STATUS[9:15]
591	193Ch	SENT_1 Channel 4 Fast message interrupt	SENT_1_FMSG_RDY[4]
592	1940h	SENT_1 Channel 4 Slow message interrupt	SENT_1_SMSG_RDY[4]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
593	1944h	SENT_1 Channel 4 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_1_CH4_STATUS[4:7] SENT_1_CH4_STATUS[9:15]
612	1990h	PSI5_0 Channel 0 DMA Combined interrupt: – DMA transfer complete – DMA SENT frame transfer complete – Fifo full or underflow of the RAM register – Underflow in the PSI5 SENT message area	PSI5_0_CH_0_DSR[IS_DMA_TF_PM_DS] PSI5_0_CH_0_DSR[IS_DMA_TF_SF] PSI5_0_CH_0_DSR[IS_DMA_PM_DS_FIFO_FULL] PSI5_0_CH_0_DSR[IS_DMA_SFUF] PSI5_0_CH_0_DSR[IS_DMA_PM_DS_UF]
613	1994h	PSI5_0 Channel 0 combined General interrupt: – New SENT message reception – CRC error – Overwrite of unread message in corresponding SENT location (in PSI5_SFR) – DPR is ready for receiving a new data – DBR is ready for receiving a new data – DSR is ready for receiving a new data – Write to DPR while it is NOT ready – Write to DBR while it is NOT ready – Write to DSR while it is NOT ready – New value in data time stamp register – New value in sync time stamp read register	PSI5_0_CH_0_GISR[IS_CESM[6:1]] PSI5_0_CH_0_GISR[IS_STS] PSI5_0_CH_0_GISR[IS_DTS] PSI5_0_CH_0_GISR[IS_DSROW] PSI5_0_CH_0_GISR[IS_BROW] PSI5_0_CH_0_GISR[DSR_RDY] PSI5_0_CH_0_GISR[DBR_RDY] PSI5_0_CH_0_GISR[DPR_RDY] PSI5_0_CH_0_GISR[IS_OWSM[6:1]] PSI5_0_CH_0_GISR[IS_NVSM[6:1]]
614	1998h	PSI5_0 Channel 0 New message combined interrupt: New data received in any enabled PSI5_0 Channel 0 message buffer	PSI5_0_CH_0_NDSR[NDS31] PSI5_0_CH_0_NDSR[NDS30] ... PSI5_0_CH_0_NDSR[NDS1] PSI5_0_CH_0_NDSR[NDS0]
615	199Ch	PSI5_0 Channel 0 Message Overwrite combined interrupt: Overwrite of any PSI5_0 Channel 0 message buffer	PSI5_0_CH_0_OWSR[NDS31] PSI5_0_CH_0_OWSR[NDS30] ... PSI5_0_CH_0_OWSR[NDS1] PSI5_0_CH_0_OWSR[NDS0]
616	19A0h	PSI5_0 Channel 0 Error combined interrupt: error in any enabled PSI5_0 Channel 0 message buffer	PSI5_0_CH_0_EISR[NDS31] PSI5_0_CH_0_EISR[NDS30] ... PSI5_0_CH_0_EISR[NDS1] PSI5_0_CH_0_EISR[NDS0]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
617	19A4h	PSI5_0 Channel 0 Global interrupt: global OR of all the previous PSI5_0 Channel 0 sources	PSI5_0_CH_0_DSR[*] PSI5_0_CH_0_GISR[*] PSI5_0_CH_0_NDSR[31:0] PSI5_0_CH_0_OWSR[31:0] PSI5_0_CH_0_EISR[31:0]
618	19A8h	PSI5_0 Channel 1 DMA Combined interrupt: – DMA transfer complete – DMA SENT frame transfer complete – Fifo full or underflow of the RAM register – Underflow in the PSI5 SENT message area	PSI5_0_CH_1_DSR[IS_DMA_TF_PM_DS] PSI5_0_CH_1_DSR[IS_DMA_TF_SF] PSI5_0_CH_1_DSR[IS_DMA_PM_DS_FIFO_FULL] PSI5_0_CH_1_DSR[IS_DMA_SFUF] PSI5_0_CH_1_DSR[IS_DMA_PM_DS_UF]
619	19ACh	PSI5_0 Channel 1 combined General interrupt: – New SENT message reception – CRC error – Overwrite of unread message in corresponding SENT location (in PSI5_SFR) – DPR is ready for receiving a new data – DBR is ready for receiving a new data – DSR is ready for receiving a new data – Write to DPR while it is NOT ready – Write to DBR while it is NOT ready – Write to DSR while it is NOT ready – New value in data time stamp register – New value in sync time stamp read register	PSI5_0_CH_1_GISR[IS_CESM[6:1]] PSI5_0_CH_1_GISR[IS_STS] PSI5_0_CH_1_GISR[IS_DTS] PSI5_0_CH_1_GISR[IS_DSROW] PSI5_0_CH_1_GISR[IS_BROW] PSI5_0_CH_1_GISR[DSR_RDY] PSI5_0_CH_1_GISR[DBR_RDY] PSI5_0_CH_1_GISR[DPR_RDY] PSI5_0_CH_1_GISR[IS_OWSM[6:1]] PSI5_0_CH_1_GISR[IS_NVSM[6:1]]
620	19B0h	PSI5_0 Channel 1 New message combined interrupt: New data received in any enabled PSI5_0 Channel 1 message buffer	PSI5_0_CH_1_NDSR[NDS31] PSI5_0_CH_1_NDSR[NDS30] ... PSI5_0_CH_1_NDSR[NDS1] PSI5_0_CH_1_NDSR[NDS0]
621	19B4h	PSI5_0 Channel 1 Message Overwrite combined interrupt: Overwrite of any PSI5_0 Channel 1 message buffer	PSI5_0_CH_1_OWSR[NDS31] PSI5_0_CH_1_OWSR[NDS30] ... PSI5_0_CH_1_OWSR[NDS1] PSI5_0_CH_1_OWSR[NDS0]
622	19B8h	PSI5_0 Channel 1 Error combined interrupt: error in any enabled PSI5_0 Channel 1 message buffer	PSI5_0_CH_1_EISR[NDS31] PSI5_0_CH_1_EISR[NDS30] ... PSI5_0_CH_1_EISR[NDS1] PSI5_0_CH_1_EISR[NDS0]
623	19BCh	PSI5_0 Channel 1 Global interrupt: global OR of all the previous PSI5_0 Channel 1 sources	PSI5_0_CH_1_DSR[*] PSI5_0_CH_1_GISR[*] PSI5_0_CH_1_NDSR[31:0] PSI5_0_CH_1_OWSR[31:0] PSI5_0_CH_1_EISR[31:0]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
624	19C0h	PSI5_1 Channel 0 DMA Combined interrupt: – DMA transfer complete – DMA SENT frame transfer complete – Fifo full or underflow of the RAM register – Underflow in the PSI5 SENT message area	PSI5_1_CH_0_DSR[IS_DMA_TF_PM_DS] PSI5_1_CH_0_DSR[IS_DMA_TF_SF] PSI5_1_CH_0_DSR[IS_DMA_PM_DS_FIFO_FULL] PSI5_1_CH_0_DSR[IS_DMA_SFUF] PSI5_1_CH_0_DSR[IS_DMA_PM_DS_UF]
625	19C4h	PSI5_1 Channel 0 combined General interrupt: – New SENT message reception – CRC error – Overwrite of unread message in corresponding SENT location (in PSI5_SFR) – DPR is ready for receiving a new data – DBR is ready for receiving a new data – DSR is ready for receiving a new data – Write to DPR while it is NOT ready – Write to DBR while it is NOT ready – Write to DSR while it is NOT ready – New value in data time stamp register – New value in sync time stamp read register	PSI5_1_CH_0_GISR[IS_CESM[6:1]] PSI5_1_CH_0_GISR[IS_STS] PSI5_1_CH_0_GISR[IS_DTS] PSI5_1_CH_0_GISR[IS_DSROW] PSI5_1_CH_0_GISR[IS_BROW] PSI5_1_CH_0_GISR[DSR_RDY] PSI5_1_CH_0_GISR[DBR_RDY] PSI5_1_CH_0_GISR[DPR_RDY] PSI5_1_CH_0_GISR[IS_OWSM[6:1]] PSI5_1_CH_0_GISR[IS_NVSM[6:1]]
626	19C8h	PSI5_1 Channel 0 New message combined interrupt: New data received in any enabled PSI5_1 Channel 0 message buffer	PSI5_1_CH_0_NDSR[NDS31] PSI5_1_CH_0_NDSR[NDS30] ... PSI5_1_CH_0_NDSR[NDS1] PSI5_1_CH_0_NDSR[NDS0]
627	19CCh	PSI5_1 Channel 0 Message Overwrite combined interrupt: Overwrite of any PSI5_1 Channel 0 message buffer	PSI5_1_CH_0_OWSR[NDS31] PSI5_1_CH_0_OWSR[NDS30] ... PSI5_1_CH_0_OWSR[NDS1] PSI5_1_CH_0_OWSR[NDS0]
628	19D0h	PSI5_1 Channel 0 Error combined interrupt: error in any enabled PSI5_1 Channel 0 message buffer	PSI5_1_CH_0_EISR[NDS31] PSI5_1_CH_0_EISR[NDS30] ... PSI5_1_CH_0_EISR[NDS1] PSI5_1_CH_0_EISR[NDS0]
629	19D4h	PSI5_1 Channel 0 Global interrupt: global OR of all the previous PSI5_1 Channel 0 sources	PSI5_1_CH_0_DSR[*] PSI5_1_CH_0_GISR[*] PSI5_1_CH_0_NDSR[31:0] PSI5_1_CH_0_OWSR[31:0] PSI5_1_CH_0_EISR[31:0]
654	1A38h	SIPI combined Error Interrupt: – Channels 0 to 3 Timeout error – Channels 0 to 3 transaction ID error – Channels 0 to 3 acknowledge error	SIPI_ERR[TOEn] SIPI_ERR[TIDEn] SIPI_ERR[ACKEn] n = [0..3]
655	1A3Ch	SIPI CRC Error interrupt	SIPI_SR[GCRCE]
656	1A40h	SIPI Channel 0 Reception combined interrupt: – Read answer received – Acknowledge received	SIPI_CSR0[RAR] SIPI_CSR0[ACKR]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
657	1A44h	SIPI Channel 1 Reception combined interrupt: – Read answer received – Acknowledge received	SIPI_CSR1[RAR] SIPI_CSR1[ACKR]
658	1A48h	SIPI Channel 2 Reception combined interrupt: – Read answer received – Acknowledge received	SIPI_CSR2[RAR] SIPI_CSR2[ACKR]
659	1A4Ch	SIPI Channel 3 Reception combined interrupt: – Read answer received – Acknowledge received	SIPI_CSR3[RAR] SIPI_CSR3[ACKR]
660	1A50h	SIPI combined Event Interrupt: – Channel 0 to 3 trigger events – Maximum count reached (streaming mode) event	SIPI_SR[TE0] SIPI_SR[TE1] SIPI_SR[TE2] SIPI_SR[TE3] SIPI_SR[MCR]
661	1A54h	LFAST_0 Transmit Complete combined interrupt: – Ping response frame transmitted – Unsolicited Frame transmitted – ICLC Frame transmitted – Data frame transmitted	LFAST_0_TISR[TXPNGF] LFAST_0_TISR[TXUNSF] LFAST_0_TISR[TXICLCF] LFAST_0_TISR[TXDTF]
662	1A58h	LFAST_0 Transmit Error combined interrupt: – Transmit interface not enabled error – Transmit data FIFO overflow error	LFAST_0_TISR[TXIEF] LFAST_0_TISR[TXOVF]
663	1A5Ch	LFAST_0 Receive combine interrupt: – Frame with CTS bit low received – Data frame received – Unsolicited Frame received	LFAST_0_RISR[RXCTSF] LFAST_0_RISR[RXDF] LFAST_0_RISR[RXUNSF]
664	1A60h	LFAST_0 Receive Error combined interrupt: – Unsolicited frame register overflow – Rx Data FIFO Minimum Threshold reached – Rx Data FIFO Max Threshold reached – Rx Data FIFO Underflow – Rx Data FIFO Overflow – Frame with unsupported frame size received – Invalid ICLC code Received – Invalid Logical Channel Type	LFAST_0_RISR[RXUOF] LFAST_0_RISR[RXMNF] LFAST_0_RISR[RXMXF] LFAST_0_RISR[RXUFF] LFAST_0_RISR[RXOFF] LFAST_0_RISR[RXSZF] LFAST_0_RISR[RXICF] LFAST_0_RISR[RXLCEF]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
665	1A64h	LFAST_0 ICLC receive combined interrupt: – Ping Frame Response failed – Ping Frame Response successful – ICLC Ping Frame Request received – ICLC frame for Test mode off received – ICLC frame for Loopback On received – ICLC frame for Clk Test mode received – ICLC frame for LFAST Slaves Tx Interface Disable received – ICLC frame for LFAST Slaves Tx Interface Enable received – ICLC frame for LFAST Slaves Rx Interface fast mode switch received – ICLC frame for LFAST Slaves Rx Interface slow mode switch received – ICLC frame for LFAST Slaves Tx Interface fast mode switch received – ICLC frame for LFAST Slaves Tx Interface slow mode switch received – ICLC frame for PLL OFF received – ICLC frame for PLL ON received	LFAST_0_RIISR[ICPFF] LFAST_0_RIISR[ICPSF] LFAST_0_RIISR[ICPRF] LFAST_0_RIISR[ICTOF] LFAST_0_RIISR[ICLPF] LFAST_0_RIISR[ICCTF] LFAST_0_RIISR[ICTDF] LFAST_0_RIISR[ICTEF] LFAST_0_RIISR[ICRFF] LFAST_0_RIISR[ICRSF] LFAST_0_RIISR[ICTFF] LFAST_0_RIISR[ICTSF] LFAST_0_RIISR[ICPOFF] LFAST_0_RIISR[ICPONF]
674	1A88h	JTAG Master combined interrupt: – Idle interrupt – Sequence Processing Unit interrupt	JTAGM_SR[SPU_INT] JTAGM_SR[Idle]
675	1A8Ch	JTAG Data Communication combined interrupt: – JDC Input data received – JDC output data sent	JDC_MSR[JIN_INT] JDC_MSR[JOUT_INT]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
677	1A94h	M_TTCAN_0 Line0 combined interrupt: Stuff Error, Format Error, Acknowledge Error, Bit Error, CRC Error, Watchdog, Bus_Off Status, Warning Status, Error Passive Status, Error Logging Overflow, Bit Error Uncorrected, Bit Error Corrected, Timeout Occurred, Unprocessed Message Discarded, Timestamp Wraparound, Tx Event FIFO Event Lost, Tx Event FIFO Full, Tx Event FIFO Watermark Reached, Tx Event FIFO New Entry, Tx FIFO Empty, Transmission Cancellation Finished, Transmission Completed, High Priority Message, Rx FIFO 1 Message Lost, Rx FIFO 1 Full, Rx FIFO 1 Watermark Reached, Rx FIFO 1 New Message, Rx FIFO 0 Message Lost, Rx FIFO 0 Full, Rx FIFO 0 Watermark Reached, Rx FIFO 0 New Message	(M_TTCAN_0_IR[0] & ~M_TTCAN_0_ILS[0]) ... (M_TTCAN_0_IR[11] & ~M_TTCAN_0_ILS[11]) (M_TTCAN_0_IR[13] & ~M_TTCAN_0_ILS[13]) ... (M_TTCAN_0_IR[31] & ~M_TTCAN_0_ILS[31])
678	1A98h	M_TTCAN_0 Line1 combined interrupt: Stuff Error, Format Error, Acknowledge Error, Bit Error, CRC Error, Watchdog, Bus_Off Status, Warning Status, Error Passive Status, Error Logging Overflow, Bit Error Uncorrected, Bit Error Corrected, Timeout Occurred, Unprocessed Message Discarded, Timestamp Wraparound, Tx Event FIFO Event Lost, Tx Event FIFO Full, Tx Event FIFO Watermark Reached, Tx Event FIFO New Entry, Tx FIFO Empty, Transmission Cancellation Finished, Transmission Completed, High Priority Message, Rx FIFO 1 Message Lost, Rx FIFO 1 Full, Rx FIFO 1 Watermark Reached, Rx FIFO 1 New Message, Rx FIFO 0 Message Lost, Rx FIFO 0 Full, Rx FIFO 0 Watermark Reached, Rx FIFO 0 New Message	(M_TTCAN_0_IR[0] & M_TTCAN_0_ILS[0]) ... (M_TTCAN_0_IR[11] & M_TTCAN_0_ILS[11]) (M_TTCAN_0_IR[13] & M_TTCAN_0_ILS[13]) ... (M_TTCAN_0_IR[31] & M_TTCAN_0_ILS[31])
679	1A9Ch	M_TTCAN_0 Register Time Mark	M_TTCAN_0_TTIR[RTMI]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
688	1AC0h	M_CAN_1 Line0 combined interrupt: Stuff Error, Format Error, Acknowledge Error, Bit Error, CRC Error, Watchdog, Bus_Off Status, Warning Status, Error Passive Status, Error Logging Overflow, Bit Error Uncorrected, Bit Error Corrected, Timeout Occurred, Unprocessed Message Discarded, Timestamp Wraparound, Tx Event FIFO Event Lost, Tx Event FIFO Full, Tx Event FIFO Watermark Reached, Tx Event FIFO New Entry, Tx FIFO Empty, Transmission Cancellation Finished, Transmission Completed, High Priority Message, Rx FIFO 1 Message Lost, Rx FIFO 1 Full, Rx FIFO 1 Watermark Reached, Rx FIFO 1 New Message, Rx FIFO 0 Message Lost, Rx FIFO 0 Full, Rx FIFO 0 Watermark Reached, Rx FIFO 0 New Message	$(M_CAN_1_IR[0] \& \sim M_CAN_1_ILS[0]) \dots (M_CAN_1_IR[11] \& \sim M_CAN_1_ILS[11]) (M_CAN_1_IR[13] \& \sim M_CAN_1_ILS[13]) \dots (M_CAN_1_IR[31] \& \sim M_CAN_1_ILS[31])$
689	1AC4h	M_CAN_1 Line1 combined interrupt: Stuff Error, Format Error, Acknowledge Error, Bit Error, CRC Error, Watchdog, Bus_Off Status, Warning Status, Error Passive Status, Error Logging Overflow, Bit Error Uncorrected, Bit Error Corrected, Timeout Occurred, Unprocessed Message Discarded, Timestamp Wraparound, Tx Event FIFO Event Lost, Tx Event FIFO Full, Tx Event FIFO Watermark Reached, Tx Event FIFO New Entry, Tx FIFO Empty, Transmission Cancellation Finished, Transmission Completed, High Priority Message, Rx FIFO 1 Message Lost, Rx FIFO 1 Full, Rx FIFO 1 Watermark Reached, Rx FIFO 1 New Message, Rx FIFO 0 Message Lost, Rx FIFO 0 Full, Rx FIFO 0 Watermark Reached, Rx FIFO 0 New Message	$(M_CAN_1_IR[0] \& M_CAN_1_ILS[0]) \dots (M_CAN_1_IR[11] \& M_CAN_1_ILS[11]) (M_CAN_1_IR[13] \& M_CAN_1_ILS[13]) \dots (M_CAN_1_IR[31] \& M_CAN_1_ILS[31])$

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
690	1AC8h	M_CAN_2 Line0 combined interrupt	(M_CAN_2_IR[0] & ~M_CAN_2_ILS[0]) ... (M_CAN_2_IR[11] & ~M_CAN_2_ILS[11]) (M_CAN_2_IR[13] & ~M_CAN_2_ILS[13]) ... (M_CAN_2_IR[31] & ~M_CAN_2_ILS[31])
691	1ACCh	M_CAN_2 Line1 combined interrupt	(M_CAN_2_IR[0] & M_CAN_2_ILS[0]) ... (M_CAN_2_IR[11] & M_CAN_2_ILS[11]) (M_CAN_2_IR[13] & M_CAN_2_ILS[13]) ... (M_CAN_2_IR[31] & M_CAN_2_ILS[31])
692	1AD0h	M_CAN_3 Line0 combined interrupt	(M_CAN_3_IR[0] & ~M_CAN_3_ILS[0]) ... (M_CAN_3_IR[11] & ~M_CAN_3_ILS[11]) (M_CAN_3_IR[13] & ~M_CAN_3_ILS[13]) ... (M_CAN_3_IR[31] & ~M_CAN_3_ILS[31])
693	1AD4h	M_CAN_3 Line1 combined interrupt	(M_CAN_3_IR[0] & M_CAN_3_ILS[0]) ... (M_CAN_3_IR[11] & M_CAN_3_ILS[11]) (M_CAN_3_IR[13] & M_CAN_3_ILS[13]) ... (M_CAN_3_IR[31] & M_CAN_3_ILS[31])
706	1B08h	GTM AEI Shared interrupt	GTM_ICM_IRQG_0[AEI_IRQ]
707	1B0Ch	GTM ARU New Data0 transferred interrupt	GTM_ICM_IRQG_0[ARU_NEW_DATA0_IRQ]
708	1B10h	GTM ARU New Data1 transferred interrupt	GTM_ICM_IRQG_0[ARU_NEW_DATA1_IRQ]
709	1B14h	GTM ARU access acknowledge interrupt	GTM_ICM_IRQG_0[ARU_ACC_ACK_IRQ]
710	1B18h	GTM broadcast shared module interrupt	GTM_ICM_IRQG_0[BRC_IRQ]
711	1B1Ch	GTM compare shared module interrupt	GTM_ICM_IRQG_0[CMR_IRQ]
712	1B20h	GTM Sensor pattern evaluation 0 interrupt	GTM_ICM_IRQG_0[SPE0_IRQ]
713	1B24h	GTM Sensor pattern evaluation 1 interrupt	GTM_ICM_IRQG_0[SPE1_IRQ]
714	1B28h	GTM Parameter Storage Module 0 Channel 0 interrupt	GTM_ICM_IRQG_0[PSM0_CH0_IRQ]
715	1B2Ch	GTM Parameter Storage Module 0 Channel 1 interrupt	GTM_ICM_IRQG_0[PSM0_CH1_IRQ]
716	1B30h	GTM Parameter Storage Module 0 Channel 2 interrupt	GTM_ICM_IRQG_0[PSM0_CH2_IRQ]
717	1B34h	GTM Parameter Storage Module 0 Channel 3 interrupt	GTM_ICM_IRQG_0[PSM0_CH3_IRQ]
718	1B38h	GTM Parameter Storage Module 0 Channel 4 interrupt	GTM_ICM_IRQG_0[PSM0_CH4_IRQ]
719	1B3Ch	GTM Parameter Storage Module 0 Channel 5 interrupt	GTM_ICM_IRQG_0[PSM0_CH5_IRQ]
720	1B40h	GTM Parameter Storage Module 0 Channel 6 interrupt	GTM_ICM_IRQG_0[PSM0_CH6_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
721	1B44h	GTM Parameter Storage Module 0 Channel 7 interrupt	GTM_ICM_IRQG_0[PSM0_CH7_IRQ]
722	1B48h	GTM DPLL Trigger Event 4 interrupt	GTM_ICM_IRQG_1[DPLL_TE4_IRQ]
723	1B4Ch	GTM DPLL Trigger Event 3 interrupt	GTM_ICM_IRQG_1[DPLL_TE3_IRQ]
724	1B50h	GTM DPLL Trigger Event 2 interrupt	GTM_ICM_IRQG_1[DPLL_TE2_IRQ]
725	1B54h	GTM DPLL Trigger Event 1 interrupt	GTM_ICM_IRQG_1[DPLL_TE1_IRQ]
726	1B58h	GTM DPLL Trigger Event 0 interrupt	GTM_ICM_IRQG_1[DPLL_TE0_IRQ]
727	1B5Ch	GTM DPLL SUB_INC2 Loss of Lock Interrupt	GTM_ICM_IRQG_1[DPLL_LL2I_IRQ]
728	1B60h	GTM DPLL SUB_INC2 Get of Lock Interrupt	GTM_ICM_IRQG_1[DPLL_GL2I_IRQ]
729	1B64h	GTM DPLL Error Interrupt	GTM_ICM_IRQG_1[DPLL_EI_IRQ]
730	1B68h	GTM DPLL SUB_INC1 Loss of Lock Interrupt	GTM_ICM_IRQG_1[DPLL_LL1I_IRQ]
731	1B6Ch	GTM DPLL SUB_INC1Get of Lock Interrupt	GTM_ICM_IRQG_1[DPLL_GL1I_IRQ]
732	1B70h	GTM DPLL Write access to RAM region 1b or 1c interrupt	GTM_ICM_IRQG_1[DPLL_W1I_IRQ]
733	1B74h	GTM DPLL Write access to RAM region 2 interrupt	GTM_ICM_IRQG_1[DPLL_W2I_IRQ]
734	1B78h	GTM DPLL Plausibility window of Tigger request violation interrupt	GTM_ICM_IRQG_1[DPLL_PW_IRQ]
735	1B7Ch	GTM DPLL Trigger active slope detection while NTI_CNT is zero interrupt	GTM_ICM_IRQG_1[DPLL_TAS_IRQ]
736	1B80h	GTM DPLL STATE active slope detected	GTM_ICM_IRQG_1[DPLL_SAS_IRQ]
737	1B84h	GTM DPLL Missing TRIGGER interrupt	GTM_ICM_IRQG_1[DPLL_MT_IRQ]
738	1B88h	GTM DPLL Missing STATE interrupt	GTM_ICM_IRQG_1[DPLL_MS_IRQ]
739	1B8Ch	GTM DPLL TRIGGER inactive slope detected interrupt	GTM_ICM_IRQG_1[DPLL_TIS_IRQ]
740	1B90h	GTM DPLL STATE inactive slope detected interrupt	GTM_ICM_IRQG_1[DPLL_SIS_IRQ]
741	1B94h	GTM DPLL Trigger maximum hold time violation interrupt	GTM_ICM_IRQG_1[DPLL_TAX_IRQ]
742	1B98h	GTM DPLL Trigger minimum hold time violation interrupt	GTM_ICM_IRQG_1[DPLL_TIN_IRQ]
743	1B9Ch	GTM DPLL enable interrupt	GTM_ICM_IRQG_1[DPLL_PE_IRQ]
744	1BA0h	GTM DPLL disable interrupt	GTM_ICM_IRQG_1[DPLL_PD_IRQ]
745	1BA4h	GTM DPLL trigger calculated duration interrupt	GTM_ICM_IRQG_1[DPLL_CDIT_IRQ]
746	1BA8h	GTM DPLL state calculated duration interrupt	GTM_ICM_IRQG_1[DPLL_CDIS_IRQ]
747	1BACH	GTM DPLL trigger out of range interrupt	GTM_ICM_IRQG_1[DPLL_TORI_IRQ]
748	1BB0h	GTM DPLL status out of range interrupt	GTM_ICM_IRQG_1[DPLL_SORI_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
749	1BB4h	GTM TIM0 Channel 0 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH0_IRQ]
750	1BB8h	GTM TIM0 Channel 1 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH1_IRQ]
751	1BBCh	GTM TIM0 Channel 2 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH2_IRQ]
752	1BC0h	GTM TIM0 Channel 3 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH3_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
753	1BC4h	GTM TIM0 Channel 4 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH4_IRQ]
754	1BC8h	GTM TIM0 Channel 5 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH5_IRQ]
755	1BCCh	GTM TIM0 Channel 6 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH6_IRQ]
756	1BD0h	GTM TIM0 Channel 7 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH7_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
757	1BD4h	GTM TIM1 Channel 0 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH0_IRQ]
758	1BD8h	GTM TIM1 Channel 1 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH1_IRQ]
759	1BDCh	GTM TIM1 Channel 2 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH2_IRQ]
760	1BE0h	GTM TIM1 Channel 3 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH3_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
761	1BE4h	GTM TIM1 Channel 4 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH4_IRQ]
762	1BE8h	GTM TIM1 Channel 5 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH5_IRQ]
763	1BEC h	GTM TIM1 Channel 6 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH6_IRQ]
764	1BF0h	GTM TIM1 Channel 7 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH7_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
765	1BF4h	GTM TIM2 Channel 0 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM2_CH0_IRQ]
766	1BF8h	GTM TIM2 Channel 1 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM2_CH1_IRQ]
767	1BFCh	GTM TIM2 Channel 2 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM2_CH2_IRQ]
768	1C00h	GTM TIM2 Channel 3 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM2_CH3_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
769	1C04h	GTM TIM2 Channel 4 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM2_CH4_IRQ]
770	1C08h	GTM TIM2 Channel 5 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM2_CH5_IRQ]
771	1C0Ch	GTM TIM2 Channel 6 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM2_CH6_IRQ]
772	1C10h	GTM TIM2 Channel 7 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM2_CH7_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
773	1C14h	GTM TIM3 Channel 0 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM3_CH0_IRQ]
774	1C18h	GTM TIM3 Channel 1 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM3_CH1_IRQ]
775	1C1Ch	GTM TIM3 Channel 2 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM3_CH2_IRQ]
776	1C20h	GTM TIM3 Channel 3 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM3_CH3_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
777	1C24h	GTM TIM3 Channel 4 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM3_CH4_IRQ]
778	1C28h	GTM TIM3 Channel 5 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM3_CH5_IRQ]
779	1C2Ch	GTM TIM3 Channel 6 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM3_CH6_IRQ]
780	1C30h	GTM TIM3 Channel 7 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Time out of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM3_CH7_IRQ]
781	1C34h	GTM MCS0 Channel 0 interrupt	GTM_ICM_IRQG_4[MCS0_CH0_IRQ]
782	1C38h	GTM MCS0 Channel 1 interrupt	GTM_ICM_IRQG_4[MCS0_CH1_IRQ]
783	1C3Ch	GTM MCS0 Channel 2 interrupt	GTM_ICM_IRQG_4[MCS0_CH2_IRQ]
784	1C40h	GTM MCS0 Channel 3 interrupt	GTM_ICM_IRQG_4[MCS0_CH3_IRQ]
785	1C44h	GTM MCS0 Channel 4 interrupt	GTM_ICM_IRQG_4[MCS0_CH4_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
786	1C48h	GTM MCS0 Channel 5 interrupt	GTM_ICM_IRQG_4[MCS0_CH5_IRQ]
787	1C4Ch	GTM MCS0 Channel 6 interrupt	GTM_ICM_IRQG_4[MCS0_CH6_IRQ]
788	1C50h	GTM MCS0 Channel 7 interrupt	GTM_ICM_IRQG_4[MCS0_CH7_IRQ]
789	1C54h	GTM MCS1 Channel 0 interrupt	GTM_ICM_IRQG_4[MCS0_CH0_IRQ]
790	1C58h	GTM MCS1 Channel 1 interrupt	GTM_ICM_IRQG_4[MCS0_CH1_IRQ]
791	1C5Ch	GTM MCS1 Channel 2 interrupt	GTM_ICM_IRQG_4[MCS0_CH2_IRQ]
792	1C60h	GTM MCS1 Channel 3 interrupt	GTM_ICM_IRQG_4[MCS0_CH3_IRQ]
793	1C64h	GTM MCS1 Channel 4 interrupt	GTM_ICM_IRQG_4[MCS0_CH4_IRQ]
794	1C68h	GTM MCS1 Channel 5 interrupt	GTM_ICM_IRQG_4[MCS0_CH5_IRQ]
795	1C6Ch	GTM MCS1 Channel 6 interrupt	GTM_ICM_IRQG_4[MCS0_CH6_IRQ]
796	1C70h	GTM MCS1 Channel 7 interrupt	GTM_ICM_IRQG_4[MCS0_CH7_IRQ]
797	1C74h	GTM MCS2 Channel 0 interrupt	GTM_ICM_IRQG_4[MCS0_CH0_IRQ]
798	1C78h	GTM MCS2 Channel 1 interrupt	GTM_ICM_IRQG_4[MCS0_CH1_IRQ]
799	1C7Ch	GTM MCS2 Channel 2 interrupt	GTM_ICM_IRQG_4[MCS0_CH2_IRQ]
800	1C80h	GTM MCS2 Channel 3 interrupt	GTM_ICM_IRQG_4[MCS0_CH3_IRQ]
801	1C84h	GTM MCS2 Channel 4 interrupt	GTM_ICM_IRQG_4[MCS0_CH4_IRQ]
802	1C88h	GTM MCS2 Channel 5 interrupt	GTM_ICM_IRQG_4[MCS0_CH5_IRQ]
803	1C8Ch	GTM MCS2 Channel 6 interrupt	GTM_ICM_IRQG_4[MCS0_CH6_IRQ]
804	1C90h	GTM MCS2 Channel 7 interrupt	GTM_ICM_IRQG_4[MCS0_CH7_IRQ]
805	1C94h	GTM MCS3 Channel 0 interrupt	GTM_ICM_IRQG_4[MCS0_CH0_IRQ]
806	1C98h	GTM MCS3 Channel 1 interrupt	GTM_ICM_IRQG_4[MCS0_CH1_IRQ]
807	1C9Ch	GTM MCS3 Channel 2 interrupt	GTM_ICM_IRQG_4[MCS0_CH2_IRQ]
808	1CA0h	GTM MCS3 Channel 3 interrupt	GTM_ICM_IRQG_4[MCS0_CH3_IRQ]
809	1CA4h	GTM MCS3 Channel 4 interrupt	GTM_ICM_IRQG_4[MCS0_CH4_IRQ]
810	1CA8h	GTM MCS3 Channel 5 interrupt	GTM_ICM_IRQG_4[MCS0_CH5_IRQ]
811	1CACh	GTM MCS3 Channel 6 interrupt	GTM_ICM_IRQG_4[MCS0_CH6_IRQ]
812	1CB0h	GTM MCS3 Channel 7 interrupt	GTM_ICM_IRQG_4[MCS0_CH7_IRQ]
813	1CB4h	GTM TOM0 Event 0 combined interrupt: – TOM0 channel 0 event – TOM0 channel 1 event	GTM_ICM_IRQG_6[TOM0_CH0_IRQ] GTM_ICM_IRQG-6[TOM0_CH1_IRQ]
814	1CB8h	GTM TOM0 Event 1 combined interrupt: – TOM0 channel 2 event – TOM0 channel 3 event	GTM_ICM_IRQG_6[TOM0_CH2_IRQ] GTM_ICM_IRQG-6[TOM0_CH3_IRQ]
815	1CBCh	GTM TOM0 Event 2 combined interrupt: – TOM0 channel 4 event – TOM0 channel 5 event	GTM_ICM_IRQG_6[TOM0_CH4_IRQ] GTM_ICM_IRQG-6[TOM0_CH5_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
816	1CC0h	GTM TOM0 Event 3 combined interrupt: – TOM0 channel 6 event – TOM0 channel 7 event	GTM_ICM_IRQG_6[TOM0_CH6_IRQ] GTM_ICM_IRQG-6[TOM0_CH7_IRQ]
817	1CC4h	GTM TOM0 Event 4 combined interrupt: – TOM0 channel 8 event – TOM0 channel 9 event	GTM_ICM_IRQG_6[TOM0_CH8_IRQ] GTM_ICM_IRQG-6[TOM0_CH9_IRQ]
818	1CC8h	GTM TOM0 Event 5 combined interrupt: – TOM0 channel 10 event – TOM0 channel 11 event	GTM_ICM_IRQG_6[TOM0_CH10_IRQ] GTM_ICM_IRQG-6[TOM0_CH11_IRQ]
819	1CCCh	GTM TOM0 Event 6 combined interrupt: – TOM0 channel 12 event – TOM0 channel 13 event	GTM_ICM_IRQG_6[TOM0_CH12_IRQ] GTM_ICM_IRQG-6[TOM0_CH13_IRQ]
820	1CD0h	GTM TOM0 Event 7 combined interrupt: – TOM0 channel 14 event – TOM0 channel 15 event	GTM_ICM_IRQG_6[TOM0_CH14_IRQ] GTM_ICM_IRQG-6[TOM0_CH15_IRQ]
821	1CD4h	GTM TOM1 Event 0 combined interrupt: – TOM0 channel 0 event – TOM0 channel 1 event	GTM_ICM_IRQG_6[TOM1_CH0_IRQ] GTM_ICM_IRQG-6[TOM1_CH1_IRQ]
822	1CD8h	GTM TOM1 Event 1 combined interrupt: – TOM0 channel 2 event – TOM0 channel 3 event	GTM_ICM_IRQG_6[TOM1_CH2_IRQ] GTM_ICM_IRQG-6[TOM1_CH3_IRQ]
823	1CDCh	GTM TOM1 Event 2 combined interrupt: – TOM0 channel 4 event – TOM0 channel 5 event	GTM_ICM_IRQG_6[TOM1_CH4_IRQ] GTM_ICM_IRQG-6[TOM1_CH5_IRQ]
824	1CE0h	GTM TOM1 Event 3 combined interrupt: – TOM0 channel 6 event – TOM0 channel 7 event	GTM_ICM_IRQG_6[TOM1_CH6_IRQ] GTM_ICM_IRQG-6[TOM1_CH7_IRQ]
825	1CE4h	GTM TOM1 Event 4 combined interrupt: – TOM0 channel 8 event – TOM0 channel 9 event	GTM_ICM_IRQG_6[TOM1_CH8_IRQ] GTM_ICM_IRQG-6[TOM1_CH9_IRQ]
826	1CE8h	GTM TOM1 Event 5 combined interrupt: – TOM0 channel 10 event – TOM0 channel 11 event	GTM_ICM_IRQG_6[TOM1_CH10_IRQ] GTM_ICM_IRQG-6[TOM1_CH11_IRQ]
827	1CECh	GTM TOM1 Event 6 combined interrupt: – TOM0 channel 12 event – TOM0 channel 13 event	GTM_ICM_IRQG_6[TOM1_CH12_IRQ] GTM_ICM_IRQG-6[TOM1_CH13_IRQ]
828	1CF0h	GTM TOM1 Event 7 combined interrupt: – TOM0 channel 14 event – TOM0 channel 15 event	GTM_ICM_IRQG_6[TOM1_CH14_IRQ] GTM_ICM_IRQG-6[TOM1_CH15_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
829	1CF4h	GTM TOM2 Event 0 combined interrupt: – TOM0 channel 0 event – TOM0 channel 1 event	GTM_ICM_IRQG_7[TOM2_CH0_IRQ] GTM_ICM_IRQG_7[TOM2_CH1_IRQ]
830	1CF8h	GTM TOM2 Event 1 combined interrupt: – TOM0 channel 2 event – TOM0 channel 3 event	GTM_ICM_IRQG_7[TOM2_CH2_IRQ] GTM_ICM_IRQG_7[TOM2_CH3_IRQ]
831	1CFCh	GTM TOM2 Event 2 combined interrupt: – TOM0 channel 4 event – TOM0 channel 5 event	GTM_ICM_IRQG_7[TOM2_CH4_IRQ] GTM_ICM_IRQG_7[TOM2_CH5_IRQ]
832	1D00h	GTM TOM2 Event 3 combined interrupt: – TOM0 channel 6 event – TOM0 channel 7 event	GTM_ICM_IRQG_7[TOM2_CH6_IRQ] GTM_ICM_IRQG_7[TOM2_CH7_IRQ]
833	1D04h	GTM TOM2 Event 4 combined interrupt: – TOM0 channel 8 event – TOM0 channel 9 event	GTM_ICM_IRQG_7[TOM2_CH8_IRQ] GTM_ICM_IRQG_7[TOM2_CH9_IRQ]
834	1D08h	GTM TOM2 Event 5 combined interrupt: – TOM0 channel 10 event – TOM0 channel 11 event	GTM_ICM_IRQG_7[TOM2_CH10_IRQ] GTM_ICM_IRQG_7[TOM2_CH11_IRQ]
835	1D0Ch	GTM TOM2 Event 6 combined interrupt: – TOM0 channel 12 event – TOM0 channel 13 event	GTM_ICM_IRQG_7[TOM2_CH12_IRQ] GTM_ICM_IRQG_7[TOM2_CH13_IRQ]
836	1D10h	GTM TOM2 Event 7 combined interrupt: – TOM0 channel 14 event – TOM0 channel 15 event	GTM_ICM_IRQG_7[TOM2_CH14_IRQ] GTM_ICM_IRQG_7[TOM2_CH15_IRQ]
837	1D14h	GTM ATOM0 Event 0 combined interrupt: – ATOM0 channel 0 event – ATOM0 channel 1 event	GTM_ICM_IRQG_9[ATOM0_CH0_IRQ] GTM_ICM_IRQG_9[ATOM0_CH1_IRQ]
838	1D18h	GTM ATOM0 Event 1 combined interrupt: – ATOM0 channel 2 event – ATOM0 channel 3 event	GTM_ICM_IRQG_9[ATOM0_CH2_IRQ] GTM_ICM_IRQG_9[ATOM0_CH3_IRQ]
839	1D1Ch	GTM ATOM0 Event 2 combined interrupt: – ATOM0 channel 4 event – ATOM0 channel 5 event	GTM_ICM_IRQG_9[ATOM0_CH4_IRQ] GTM_ICM_IRQG_9[ATOM0_CH5_IRQ]
840	1D20h	GTM ATOM0 Event 3 combined interrupt: – ATOM0 channel 6 event – ATOM0 channel 7 event	GTM_ICM_IRQG_9[ATOM0_CH6_IRQ] GTM_ICM_IRQG_9[ATOM0_CH7_IRQ]
841	1D24h	GTM ATOM1 Event 0 combined interrupt: – ATOM1 channel 0 event – ATOM1 channel 1 event	GTM_ICM_IRQG_9[ATOM1_CH0_IRQ] GTM_ICM_IRQG_9[ATOM1_CH1_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
842	1D28h	GTM ATOM1 Event 1 combined interrupt: – ATOM1 channel 2 event – ATOM1 channel 3 event	GTM_ICM_IRQG_9[ATOM1_CH2_IRQ] GTM_ICM_IRQG_9[ATOM1_CH3_IRQ]
843	1D2Ch	GTM ATOM1 Event 2 combined interrupt: – ATOM1 channel 4 event – ATOM1 channel 5 event	GTM_ICM_IRQG_9[ATOM1_CH4_IRQ] GTM_ICM_IRQG_9[ATOM1_CH5_IRQ]
844	1D30h	GTM ATOM1 Event 3 combined interrupt: – ATOM1 channel 6 event – ATOM1 channel 7 event	GTM_ICM_IRQG_9[ATOM1_CH6_IRQ] GTM_ICM_IRQG_9[ATOM1_CH7_IRQ]
845	1D34h	GTM ATOM2 Event 0 combined interrupt: – ATOM2 channel 0 event – ATOM2 channel 1 event	GTM_ICM_IRQG_9[ATOM2_CH0_IRQ] GTM_ICM_IRQG_9[ATOM2_CH1_IRQ]
846	1D38h	GTM ATOM2 Event 1 combined interrupt: – ATOM2 channel 2 event – ATOM2 channel 3 event	GTM_ICM_IRQG_9[ATOM2_CH2_IRQ] GTM_ICM_IRQG_9[ATOM2_CH3_IRQ]
847	1D3Ch	GTM ATOM2 Event 2 combined interrupt: – ATOM2 channel 4 event – ATOM2 channel 5 event	GTM_ICM_IRQG_9[ATOM2_CH4_IRQ] GTM_ICM_IRQG_9[ATOM2_CH5_IRQ]
848	1D40h	GTM ATOM2 Event 3 combined interrupt: – ATOM2 channel 6 event – ATOM2 channel 7 event	GTM_ICM_IRQG_9[ATOM2_CH6_IRQ] GTM_ICM_IRQG_9[ATOM2_CH7_IRQ]
849	1D44h	GTM ATOM3 Event 0 combined interrupt: – ATOM3 channel 0 event – ATOM3 channel 1 event	GTM_ICM_IRQG_9[ATOM3_CH0_IRQ] GTM_ICM_IRQG_9[ATOM3_CH1_IRQ]
850	1D48h	GTM ATOM3 Event 1 combined interrupt: – ATOM3 channel 2 event – ATOM3 channel 3 event	GTM_ICM_IRQG_9[ATOM3_CH2_IRQ] GTM_ICM_IRQG_9[ATOM3_CH3_IRQ]
851	1D4Ch	GTM ATOM3 Event 2 combined interrupt: – ATOM3 channel 4 event – ATOM3 channel 5 event	GTM_ICM_IRQG_9[ATOM3_CH4_IRQ] GTM_ICM_IRQG_9[ATOM3_CH5_IRQ]
852	1D50h	GTM ATOM3 Event 3 combined interrupt: – ATOM3 channel 6 event – ATOM3 channel 7 event	GTM_ICM_IRQG_9[ATOM3_CH6_IRQ] GTM_ICM_IRQG_9[ATOM3_CH7_IRQ]
853	1D54h	GTM ATOM4 Event 0 combined interrupt: – ATOM4 channel 0 event – ATOM4 channel 1 event	GTM_ICM_IRQG_10[ATOM4_CH0_IRQ] GTM_ICM_IRQG_10[ATOM4_CH1_IRQ]
854	1D58h	GTM ATOM4 Event 1 combined interrupt: – ATOM4 channel 2 event – ATOM4 channel 3 event	GTM_ICM_IRQG_10[ATOM4_CH2_IRQ] GTM_ICM_IRQG_10[ATOM4_CH3_IRQ]

Table 40. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
855	1D5Ch	GTM ATOM4 Event 2 combined interrupt: – ATOM4 channel 4 event – ATOM4 channel 5 event	GTM_ICM_IRQG_10[ATOM4_CH4_IRQ] GTM_ICM_IRQG_10[ATOM4_CH5_IRQ]
856	1D60h	GTM ATOM4 Event 3 combined interrupt: – ATOM4 channel 6 event – ATOM4 channel 7 event	GTM_ICM_IRQG_10[ATOM4_CH6_IRQ] GTM_ICM_IRQG_10[ATOM4_CH7_IRQ]
931	1E8Ch	GTM Error IRQ	GTM_ERR_IRQ
946	1EC8h	PSI5-S combined interrupt: – Mailbox status interrupt 0 – Error statusinterrupt 0	PS_MBOX_SR_IRQ[0] PS_ERR_SR_IRQ[0]
947	1ECCh	PSI5-S combined interrupt: – Mailbox status interrupt 1 – Error statusinterrupt 1	PS_MBOX_SR_IRQ[1] PS_ERR_SR_IRQ[1]
948	1ED0h	PSI5-S combined interrupt: – Mailbox status interrupt 2 – Error statusinterrupt 2	PS_MBOX_SR_IRQ[2] PS_ERR_SR_IRQ[2]
949	1ED4h	PSI5-S combined interrupt: – Mailbox status interrupt 3 – Error statusinterrupt 3	PS_MBOX_SR_IRQ[3] PS_ERR_SR_IRQ[3]
950	1ED8h	PSI5-S combined interrupt: – Mailbox status interrupt 4 – Error statusinterrupt 4	"PS_MBOX_SR_IRQ[4] PS_ERR_SR_IRQ[4]
951	1EDCh	PSI5-S combined interrupt: – Mailbox status interrupt 5 – Error statusinterrupt 5	PS_MBOX_SR_IRQ[5] PS_ERR_SR_IRQ[5]
952	1EE0h	PSI5-S combined interrupt: – Mailbox status interrupt 6 – Error statusinterrupt 6	PS_MBOX_SR_IRQ[6] PS_ERR_SR_IRQ[6]
953	1EE4h	PSI5-S combined interrupt: – Mailbox status interrupt 7 – Error statusinterrupt 7	PS_MBOX_SR_IRQ[7] PS_ERR_SR_IRQ[7]
954	1EE8h	PSI5-S ECU to sensor interrupt 1	PSI5_E2SSR[1]
955	1EECh	PSI5-S ECU to sensor interrupt 2	PSI5_E2SSR[2]
956	1EF0h	PSI5-S ECU to sensor interrupt 3	PSI5_E2SSR[3]
957	1EF4h	PSI5-S ECU to sensor interrupt 4	PSI5_E2SSR[4]
958	1EF8h	PSI5-S ECU to sensor interrupt 5	PSI5_E2SSR[5]
959	1EFCh	PSI5-S ECU to sensor interrupt 6	PSI5_E2SSR[6]
960	1F00h	PSI5-S ECU to sensor interrupt 7	PSI5_E2SSR[7]
961	1F04h	MRU data overwrite global error interrupt	PS_GLSR

1. This character is used to denote the OR operator throughout this table.

7.3.7.4 Interrupt Latency

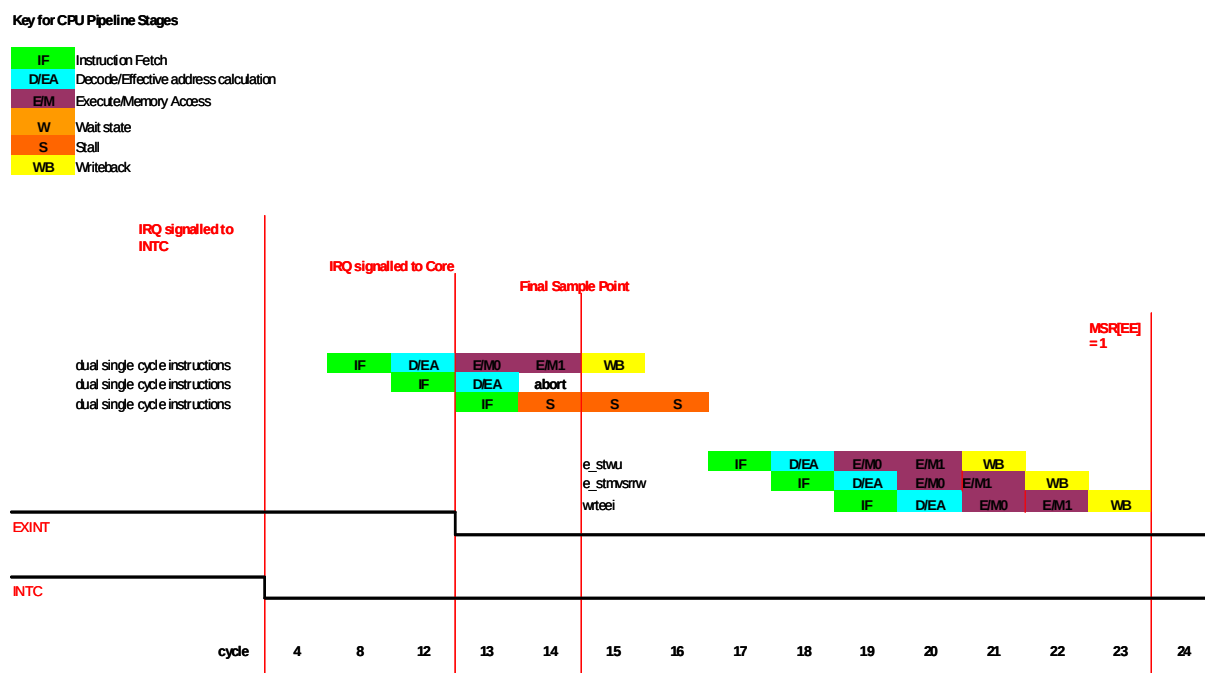
The Interrupt systems is designed to provide a very short latency between the assertion of an interrupt request from a peripheral and the first instruction of the Interrupt Service Routine.

The latency time is difficult to define exactly because it depends what interrupts the INTC is processing and what the CPU is doing.

But in general, the latency time is divided into 3 sub-sections:

- INTC Latency
 - 3 clock cycles (INTC clock domain) from the assertion of the interrupt request from the peripheral to assertion of the interrupt request to the CPU
- CPU Latency
 - 4 clock cycles (CPU clock domain) from the assertion of the interrupt request from the INTC to the execution of the first instruction
- Software Latency
 - 7 clock cycles (CPU clock domain) to execute the 3 instructions needed to save essential registers and enable interrupts

Figure 19. Interrupt timing



7.3.7.5 INTC (hsm_z0_platform) configuration

For interrupts specific to the Hardware Security Module (HSM) please refer to the *SPC57EM80xx Security Reference Manual*.

7.3.8 DMA Controller configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

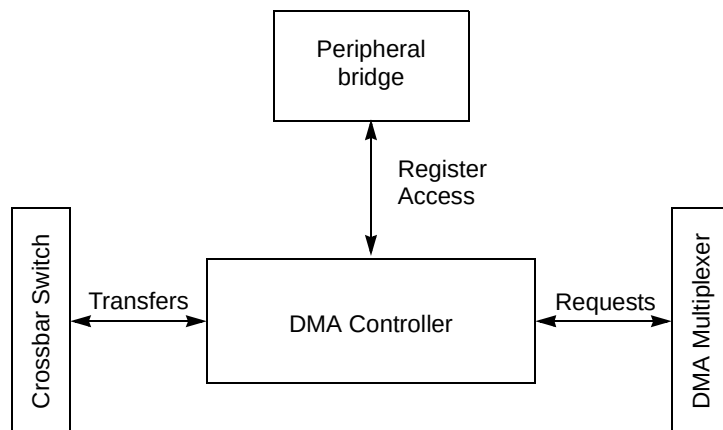


Figure 20. DMA controller configuration

Table 41. Reference links to related information

Topic	Related module	Reference
Full description	DMA Controller	Chapter 25: Enhanced Direct Memory Access (eDMA)
System memory map	—	Chapter 5: Memory Map
Register access	Peripheral bridge	Chapter 20: Peripheral Bridge
Clocking	—	Chapter 27: Clocking
Transfers	Crossbar switch	Chapter 17: Crossbar Switch (XBAR)

7.3.9 DMACHMUX configuration

The peripheral DMA requests are serviced by a DMA controller. A static multiplexing scheme allows the support of more peripheral requests than available DMA channels.

7.3.9.1 DMA channel assignment

SPC57EM80xx has six multiplexers. [Table 42](#) lists which channels apply to each multiplexer.

Table 42. SPC57EM80xx multiplexer and channel list

DMAMUX instance	DMA channel
0	0–7
1	8–15
2	16–23
3	24–31

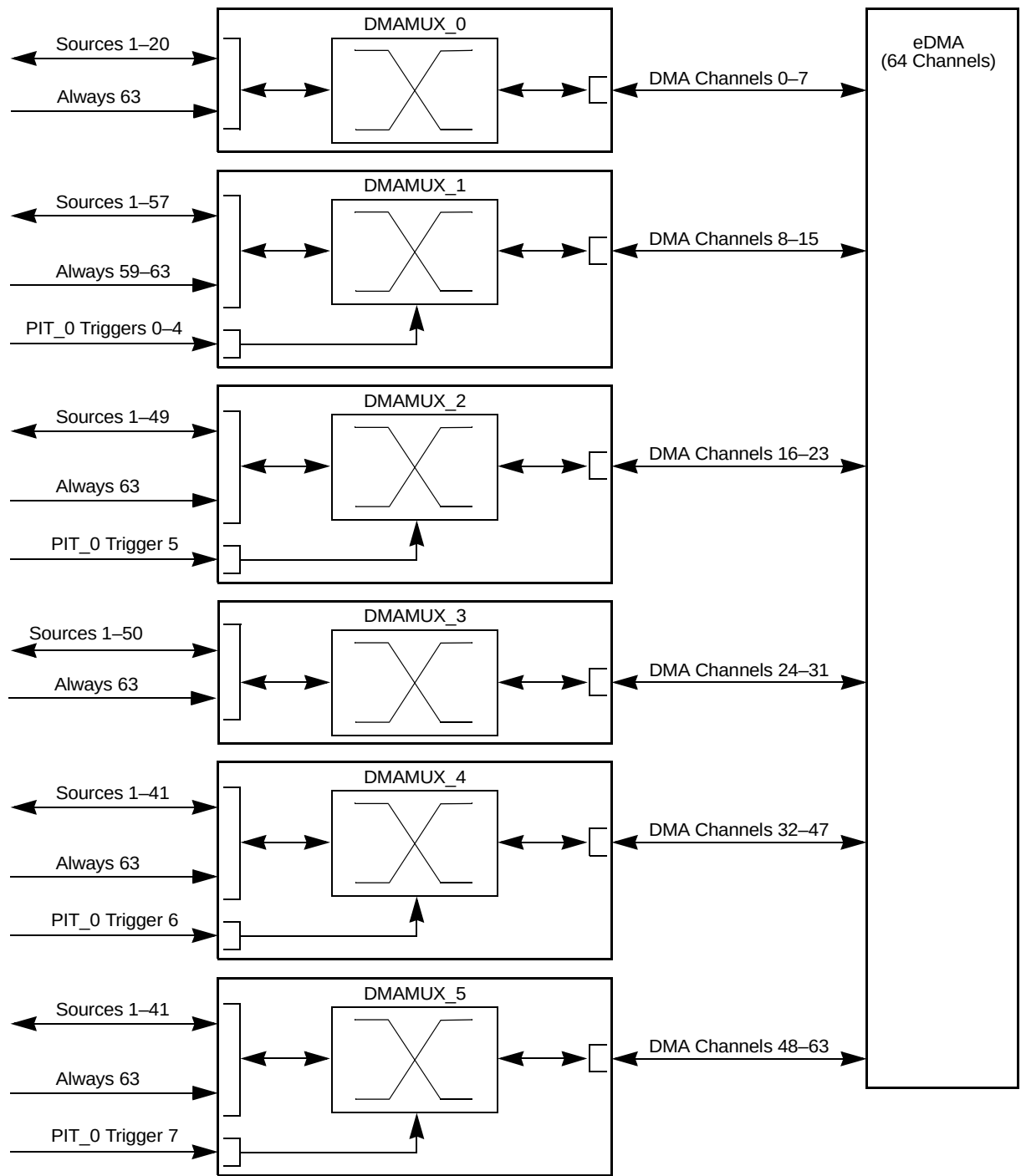
Table 42. SPC57EM80xx multiplexer and channel list

DMAMUX instance	DMA channel
4	32–47
5	48–63

7.3.9.2 DMA channel multiplexing block diagram

The connections of the DMA channel multiplexors to the eDMA are shown in [Figure 21](#). For the lower thirty-two channels, One DMAMUX is instantiated for every eight DMA channels. For the upper thirty-two channels, one DMAMUX is instantiated for every sixteen DMA channels.

Figure 21. DMA channel multiplexing



7.3.9.3 DMA request mapping

The mapping of the peripheral DMA requests to the DMA_CH_MUX input sources is shown in [Table 43](#).

Table 43. DMAMUX peripheral DMA request to input source mapping

Source	Peripheral DMA requests					
	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3	DMAMUX_4	DMAMUX_5
0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
1	ADC_SAR_0 EOC	DSPI_12 RX	ADC_SAR_2 EOC	ADC_SAR_3 EOC	ADC_SAR_4 EOC	ADC_SAR_7 EOC
2	ADC_SAR_1 EOC	DSPI_12 TX	ADC_SD_1 EOC	DSPI_2 RX	ADC_SAR_6 EOC	ADC_SD_4 EOC
3	ADC_SAR_B EOC	LINFlex_0 RX	DSPI_1 RX	DSPI_2 TX	ADC_SD_2 EOC	ADC_SD_5 EOC
4	ADC_SD_0 EOC	LINFlex_0 TX	DSPI_1 TX	LINFlex_2 RX	ADC_SD_3 EOC	DSPI_5 RX
5	DSPI_0 RX	LINFlex_1 RX	SENT_1 RX FAST	LINFlex_2 TX	DSPI_3 RX	DSPI_5 TX
6	DSPI_0 TX	LINFlex_1 TX	SENT_1 RX SLOW	I2C_0 RX	DSPI_3 TX	LINFlex_1 RX
7	DSPI_4 RX	LINFlex_14 RX	PSI5_0 CH0 RX PSI5	I2C_0 TX	LINFlex_0 RX	LINFlex_1 TX
8	DSPI_4 Tx	LINFlex_14 TX	PSI5_0 CH0 RX SENT	PSI5_1 CH0 RX PSI5	LINFlex_0 TX	LINFlex_15 RX
9	Reserved	SENT_0 RX FAST	SIUL2 REQ2	PSI5_1 CH0 RX SENT	LINFlex_14 RX	LINFlex_15 TX
10	ADC_SAR_4 EOC	SENT_0 RX SLOW	SIUL2 REQ4	SIUL2 REQ5	LINFlex_14 TX	SENT_0 RX FAST
11	ADC_SD_3 EOC	SIPI CH0	GTM_PSM0_IRQ0	GTM_PSM0_IRQ4	PSI5_0 CH1 RX PSI5	SENT_0 RX SLOW
12	M_CAN_1	SIPI CH1	GTM_PSM0_IRQ1	GTM_PSM0_IRQ5	PSI5_0 CH1 RX SENT	SIPI CH2
13	M_CAN_2	SIPI CH2	GTM_PSM0_IRQ2	GTM_PSM0_IRQ6	SIPI CH0	SIPI CH3
14	SENT_0 RX FAST	SIPI CH3	GTM_PSM0_IRQ3	GTM_PSM0_IRQ7	SIPI CH1	SIUL2 REQ10
15	SENT_0 RX SLOW	SIUL2 REQ0	GTM_TIM1_IRQ0	GTM_TIM1_IRQ4	SIUL2 REQ9	GTM_TIM0_IRQ2
16	LINFlex_0 RX	SIUL2 REQ1	GTM_TIM1_IRQ1	GTM_TIM1_IRQ5	GTM_TIM0_IRQ0	GTM_TIM0_IRQ3
17	LINFlex_0 TX	GTM_TIM0_IRQ0	GTM_TIM1_IRQ2	GTM_TIM1_IRQ6	GTM_TIM0_IRQ1	GTM_TIM0_IRQ4
18	LINFlex_14 RX	GTM_TIM0_IRQ1	GTM_TIM1_IRQ3	GTM_TIM1_IRQ7	GTM_TIM0_IRQ0	GTM_TIM0_IRQ5
19	DSPI_0 CMD	GTM_TIM0_IRQ2	GTM_TIM1_IRQ0	GTM_TIM1_IRQ4	GTM_TIM0_IRQ1	GTM_PSM0_IRQ4
20	DSPI_4 CMD	GTM_TIM0_IRQ3	GTM_TIM1_IRQ1	GTM_TIM1_IRQ5	GTM_PSM0_IRQ0	GTM_PSM0_IRQ5
21	Reserved	GTM_TIM0_IRQ4	GTM_TIM1_IRQ2	GTM_TIM1_IRQ6	GTM_PSM0_IRQ1	GTM_PSM0_IRQ6
22	Reserved	GTM_TIM0_IRQ5	GTM_TIM1_IRQ3	GTM_TIM1_IRQ7	GTM_PSM0_IRQ2	GTM_PSM0_IRQ7
23	Reserved	GTM_TIM0_IRQ6	GTM_ATOM1_IRQ0	GTM_ATOM1_IRQ2	GTM_PSM0_IRQ3	GTM_TIM1_IRQ4
24	Reserved	GTM_TIM0_IRQ7	GTM_ATOM1_IRQ1	GTM_ATOM1_IRQ3	GTM_TIM1_IRQ0	GTM_TIM1_IRQ5
25	Reserved	GTM_TIM0_IRQ0	GTM_MCS1_IRQ0	GTM_MCS1_IRQ4	GTM_TIM1_IRQ1	GTM_TIM3_IRQ4
26	Reserved	GTM_TIM0_IRQ1	GTM_MCS1_IRQ1	GTM_MCS1_IRQ5	GTM_TIM3_IRQ0	GTM_TIM3_IRQ5
27	Reserved	GTM_TIM0_IRQ2	GTM_MCS1_IRQ2	GTM_MCS1_IRQ6	GTM_TIM3_IRQ1	GTM_TIM3_IRQ6
28	Reserved	GTM_TIM0_IRQ3	GTM_MCS1_IRQ3	GTM_MCS1_IRQ7	GTM_TIM3_IRQ2	GTM_TIM3_IRQ7
29	Reserved	GTM_TIM0_IRQ4	GTM_TIM2_IRQ0	GTM_TIM2_IRQ4	GTM_TIM3_IRQ3	GTM_MCS3_IRQ4
30	Reserved	GTM_TIM0_IRQ5	GTM_TIM2_IRQ1	GTM_TIM2_IRQ5	GTM_MCS3_IRQ0	GTM_MCS3_IRQ5
31	Reserved	GTM_TIM0_IRQ6	GTM_TIM2_IRQ2	GTM_TIM2_IRQ6	GTM_MCS3_IRQ1	GTM_MCS3_IRQ6
32	Reserved	GTM_TIM0_IRQ7	GTM_TIM2_IRQ3	GTM_TIM2_IRQ7	GTM_MCS3_IRQ2	GTM_MCS3_IRQ7
33	Reserved	GTM_ATOM0_IRQ0	GTM_ATOM2_IRQ0	GTM_ATOM2_IRQ2	GTM_MCS3_IRQ3	DSPI_5 CMD
34	Reserved	GTM_ATOM0_IRQ1	GTM_ATOM2_IRQ1	GTM_ATOM2_IRQ3	DSPI_3 CMD	M_CAN_1

Table 43. DMAMUX peripheral DMA request to input source mapping (continued)

Source	Peripheral DMA requests					
	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3	DMAMUX_4	DMAMUX_5
35	Reserved	GTM_ATOM0_IRQ2	GTM_MCS2_IRQ0	GTM_MCS2_IRQ4	ADC_SD_1 EOC	M_CAN_2
36	Reserved	GTM_ATOM0_IRQ3	GTM_MCS2_IRQ1	GTM_MCS2_IRQ5	ADC_SD_4 EOC	DSPI_3 RX
37	Reserved	GTM_MCS0_IRQ0	GTM_MCS2_IRQ2	GTM_MCS2_IRQ6	ADC_SD_5 EOC	DSPI_3 TX
38	Reserved	GTM_MCS0_IRQ1	GTM_MCS2_IRQ3	GTM_MCS2_IRQ7	ADC_SAR_0 EOC	ADC_SAR_3 EOC
39	Reserved	GTM_MCS0_IRQ2	GTM_ATOM3_IRQ0	GTM_ATOM3_IRQ2	DSPI_0 CMD	LINFlex_2 RX
40	Reserved	GTM_MCS0_IRQ3	GTM_ATOM3_IRQ1	GTM_ATOM3_IRQ3	DSPI_0 RX	LINFlex_2 TX
41	Reserved	GTM_MCS0_IRQ4	ADC_SD_2 EOC	SIUL2 REQ8	DSPI_0 TX	ADC_SAR_1 EOC
42	Reserved	GTM_MCS0_IRQ5	DSPI_1 CMD	Reserved	Reserved	Reserved
43	Reserved	GTM_MCS0_IRQ6	DSPI_2 RX	ADC_SD_3 EOC	Reserved	Reserved
44	Reserved	GTM_MCS0_IRQ7	DSPI_2 TX	ADC_SAR_6 EOC	Reserved	Reserved
45	Reserved	LINFlex_15 RX	LINFlex_2 RX	DSPI_2 CMD	Reserved	Reserved
46	Reserved	LINFlex_15 TX	LINFlex_2 TX	DSPI_1 RX	Reserved	Reserved
47	Reserved	DSPI_5 RX	GTM_SPE0	DSPI_1 TX	Reserved	Reserved
48	Reserved	DSPI_5 TX	GTM_SPE1	ADC_SAR_2 EOC	Reserved	Reserved
49	Reserved	DSPI_5 CMD	PSI5-S	ADC_SD_2 EOC	Reserved	Reserved
50	Reserved	DSPI_12 CMD	Reserved	PSI5-S	Reserved	Reserved
51	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
52	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
53	Reserved	DSPI_0 RX	Reserved	Reserved	Reserved	Reserved
54	Reserved	DSPI_0 TX	Reserved	Reserved	Reserved	Reserved
55	Reserved	ADC_SAR_0 EOC	Reserved	Reserved	Reserved	Reserved
56	Reserved	ADC_SAR_4 EOC	Reserved	Reserved	Reserved	Reserved
57	Reserved	ADC_SD_3 EOC	Reserved	Reserved	Reserved	Reserved
58	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
59	Reserved	Always on	Reserved	Reserved	Reserved	Reserved
60	Reserved	Always on	Reserved	Reserved	Reserved	Reserved
61	Reserved	Always on	Reserved	Reserved	Reserved	Reserved
62	Reserved	Always on	Reserved	Reserved	Reserved	Reserved
63	Always on	Always on	Always on	Always on	Always on	Always on

Note: Just as multiple CHCONFIG registers within the same instance must not select the same source value, CHCONFIG registers among multiple instances must not select the same source value.

The mapping of the PIT periodic triggers to the DMA channel sources is shown in [Table 44](#).

Table 44. PIT_0 periodic trigger to DMA channel mapping

PIT_0 periodic trigger	DMA channel
0	8
1	9
2	10
3	11
4	12
5	16
6	32
7	48

7.3.9.4 DMAMUX Memory map

The 6 instances of DMAMUX (0–5) share one PBRIDGE slot. Each instance is allocated 512 bytes of memory space. The DMAMUX address offsets are shown in [Table 45](#).

Table 45. DMAMUX memory map

Offset ⁽¹⁾	Register ⁽²⁾	Access
0x0000	DMAMUX_0 CHCONFIG0–CHCONFIG7 — Channel configuration registers	R/W
0x0200	DMAMUX_1 CHCONFIG0–CHCONFIG7 — Channel configuration registers	R/W
0x0400	DMAMUX_2 CHCONFIG0–CHCONFIG7 — Channel configuration registers	R/W
0x0600	DMAMUX_3 CHCONFIG0–CHCONFIG7 — Channel configuration registers	R/W
0x0800	DMAMUX_4 CHCONFIG0–CHCONFIG15 — Channel configuration registers	R/W
0x0A00	DMAMUX_5 CHCONFIG0–CHCONFIG15 — Channel configuration registers	R/W

1. Offsets are from DMAMUX base address (0xFFF6_C000), shown in [Chapter 5: Memory Map](#).
2. Each channel must be configured separately via settings in a dedicated configuration channel. There is one configuration register per channel for each DMAMUX. See the DMACHMUX chapter for details.

Table 46. Reference links to related information

Topic	Related module	Reference
Full description	DMA Channel Multiplexer	Chapter 26: DMACHMUX
DMA controller description	DMA Controller	Chapter 25: Enhanced Direct Memory Access (eDMA)
System memory map	—	Chapter 5: Memory Map

7.4 Clocking

See [Chapter 27: Clocking](#) for a description of the architecture for the system level clocks including clock generation, clock sources, and peripheral clocks. Also see [Chapter 29: Clock Monitor Unit \(CMU\)](#) for the CMU configuration details.

7.5 Memories and memory interfaces

See the chapters shown in [Table 47](#) for detailed information on memories and memory interfaces.

Table 47. Reference links to related information

Topic	Related module	Reference
Embedded memories	—	Chapter 3: Embedded memories
System memory map	—	Chapter 5: Memory Map
Platform Flash	—	Chapter 34: Flash memory controller
Platform RAM Controller	—	Chapter 33: RAM controller (PRAM)
Embedded flash memory	—	Chapter 35: Embedded Flash Memory (c55fmc)
Decorated Storage Memory Controller	—	Chapter 36: Decorated Storage Memory Controller (DSMC)

7.5.1 Flash memory controller (PFLASH) configuration

The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the crossbar switch. The configuration and control functions are accessed via control registers, which are read/write accessible only in supervisor mode.

Three of the registers, PFCR1, PFCR2, and PFAPR, control the interaction of master modules with the flash array by enabling/disabling prefetch or by controlling access to the flash array on a per-master basis. Use of the fields in these registers requires knowledge of the number assigned to each master. For example, the PFAPR[M2AP] field controls flash array access by master 2, the PFAPR[M3AP] field controls flash array access by master 3, and so on. See [Table 28 \(Flash memory port assignments\)](#) in the crossbar switch configuration section for details. See the PFCR3 register details in the flash memory controller chapter for more details on configuring the flash controller.

Table 48. Reference links to related information

Topic	Related module	Reference
Full description	Flash Memory Controller (PFLASH)	Chapter 34: Flash memory controller
Flash memory architecture overview	Flash memory, Flash controller	Section 3.3, Embedded flash memory
Flash memory	Flash memory	Chapter 35: Embedded Flash Memory (c55fmc)
Flash port assignments	Crossbar switch	Section 7.3.3, Crossbar switch configuration
System memory map	—	Chapter 5: Memory Map

The master assignments are shown in [Table 49](#). Note that these are the same master assignments used elsewhere in the chip, for example, the SMPU.

Table 49. Flash controller bus master assignments

Bus Master Number	Bus Master
0	Core0
1	Core1
2	Core2
3	DMA
4	Ethernet
5	FlexRay
6	Reserved
7	SIPI/DigiRF
8	Core0 Debug
9	Core1 Debug
10	Core2 Debug
11	Reserved
12	Reserved
13	Reserved
14	HSM
15	Reserved

7.5.1.1 Calibration remapping support

The PFLASH_C55FM flash memory controller supports calibration development by providing a remapping function to route flash accesses to overlay RAM or on-chip system RAM that can be used as overlay RAM. The relationship between flash access latency and remapped overlay RAM latency is contingent upon operating frequency and intrinsic memory access times. [Table 49](#) describes the intrinsic cycle latency on a random, initial access for each supported overlay RAM target.

(See [Section 34.5.14, PFlash calibration remap support](#) for more details on the relationship between flash access latency and overlay RAM access latency.

Note: Cycle times measured based on the PFLASH_C55FM clock. See [Chapter 27: Clocking](#) for more details on clocking.

Table 50. Calibration memory access times

Calibration memory	Burst access time (cycles)
On-chip overlay RAM	5
Buddy device overlay RAM	8
System RAM	9

7.5.1.2 Wait state settings for overlay flash latency matching

A remapped calibration load access has a fixed, intrinsic 4-beat latency, where the access time of each individual beat depends on the overlay target - system RAM, on-chip overlay RAM or buddy device. Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time for a flash access as well as the required access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. [Table 51](#) describes the minimum supported flash wait state setting to guarantee a remapped access matches the response latency of a flash access.

Table 51. Minimum wait state settings for overlay flash latency match

Overlay Target	PFCR1[RWSC] Minimum Value
BD RAM	6
On-Chip Overlay RAM	3
System RAM	6

If PFCR1[RWSC] is set sufficiently high such that the remapped burst access completes before flash read data is expected to be returned, the flash controller stalls the return of overlay data the appropriate number of additional cycles to ensure the access time matches the response latency of a flash access. Once the overlay access is stored in the flash controller mini-cache, subsequent requests that hit in the mini-cache are returned with zero wait state latency.

7.6 Analog modules

This section highlights the following analog modules that are implemented on SPC57EM80xx:

Table 52. Analog modules

Module or Subsystem	Device-specific information	Block details
Temperature sensor	Section 7.6.1, Temperature sensor configuration	Chapter 41: Temperature Sensor
ADC subsystem	Section 38.1, ADC overview	See individual modules.
Sigma-Delta Analog-to-Digital Converters (SDADCs)	Section 38.2.1, Sigma-Delta Analog-to-Digital Converter (SDADC)	Chapter 39: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface
Successive Approximation Register Analog-to-Digital Converters (SARADCs)	Section 38.2.2, Successive Approximation Register Analog-to-Digital Converter (SARADC)	Chapter 40: Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface

7.6.1 Temperature sensor configuration

The temperature sensor is used to measure the temperature of the SPC57EM80xx device. The temperature sensor has the following outputs:

- An analog voltage that is linearly proportional to temperature.
- Three digital outputs that signal under- or over-temperature operating conditions.

During production testing of the device, the output of the temperature sensor is sampled by SARB input channel 120 of the onboard ADC module at a predefined high temperature and also at a predefined low temperature. These two temperature sensor readings are used to calculate two constants, TSCA and TSCB, according to the formulas in [Chapter 41: Temperature Sensor](#).

Once calculated, the values for TSCA and TSCB are programmed during production test into the UTEST flash memory at locations 0x0040_0000 and 0x0040_0002, respectively.

The ADC bandgap voltage, used in the temperature calculation, is sampled by SARB input channel 121 of the onboard ADC module.

Software must perform the following steps in order to calculate device temperature.

1. Measure the bandgap voltage, $V_{BG}(T)$, using SARB input channel 121 of the onboard ADC module.
2. Calculate $V_{BG_CODE}(T)$ according to the formula in [Chapter 41: Temperature Sensor](#).
3. Measure the temperature sensor output voltage, $V_{TSENS}(T)$, using SARB input channel 120 of the onboard ADC module.
4. Calculate $T_{TSENS_CODE}(T)$ according to the formula in [Chapter 41: Temperature Sensor](#).
5. Retrieve parameters TSCA and TSCB from flash memory.
6. Calculate T according to the formula in [Chapter 41: Temperature Sensor](#).

7.6.2 Analog to Digital Converters (ADC) configuration

See [Chapter 38: Analog-to-Digital Converters \(ADC\) Configuration](#), for an overview of the ADC subsystem, including a block diagram and a summary of analog input pin multiplexing. The chapter also provides configuration details for the SARADCs and SDADCs.

7.6.2.1 SDADC configuration

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides the accurate conversion data for a wide range of applications.

SPC57EM80xx includes six independent 16-bit SDADCs.

For details, see [Chapter 38: Analog-to-Digital Converters \(ADC\) Configuration](#).

7.6.2.2 SARADC configuration

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for an application. The SARADC provides accurate and fast conversion data for a wide range of applications. Each SARADC analog block has a corresponding digital interface.

SPC57EM80xx includes eight (8) independent 12-bit SARADCs. SAR ADCB has access to all external analog input pins, access to the external multiplexed inputs, and has special safety features.

For details, see [Chapter 38: Analog-to-Digital Converters \(ADC\) Configuration](#).

7.7 Timers

Three types of timers are implemented on SPC57EM80xx and a complex timer subsystem.

Table 53. Timers

Timer	Device-specific information	Block details
System Timer Module (STM)	Section 7.7.1, System Timer Module (STM) configuration	Chapter 43: System Timer Module (STM)
Software Watchdog Timer (SWT)	Section 7.7.2, Software Watchdog Timer (SWT) configuration	Chapter 44: Software Watchdog Timer (SWT)
Periodic Interrupt Timer (PIT)	Section 7.7.3, PIT configuration	Chapter 45: Periodic Interrupt Timer (PIT)
GTM103 Wrapper Module Configuration	Section 7.7.4, GTM 103 Integration Module (GTMINT103) configuration	Chapter 88: GTM103 Integration (GTMINT) Module

Each core will have a dedicated System Timer Module (STM) and a dedicated Software Watchdog Timer (SWT). The checker core does not have a dedicated STM.

The Secure SWT is on the Hardware Security Module (HSM). It is an enhanced version of the SWT that has the ability to trigger an interrupt not only based on some number of clock cycles, but also based on code executing at some user-defined address. For more information, refer to the *SPC57EM80xx Security Manual*.

Two Periodic Interrupt Timer (PIT) module instances are implemented. One PIT module is for 64-bit timer implementation by way of two 32-bit timer channels (0 and 1) with chaining mode; the timer value is unchanged by a non-destructive reset. The other PIT module is for general use and has eight 32-bit timer channels with no chaining mode.

The GTM103 is an implementation of the Robert Bosch GmbH GTM timer subsystem. It is not a single module; it is a complex timer subsystem that consists of many modules. It can be used to implement highly complex timer functions.

7.7.1 System Timer Module (STM) configuration

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the appropriate system clock divided by an 8-bit prescale value (1 to 256).

SPC57EM80xx includes three STMs, one for each core, with four 32-bit compare channels in each STM.

Table 54. System Timer Module (STM) instances

Instance	Description	Number of Channels	Number of Registers (32-bit)
STM0	Intended for Main Core_0 in the high-speed computational clock domain.	4	14
STM1	Intended for Main Core_1 in the high-speed computational clock domain.	4	14
STM2	Intended for Peripheral Core_2 in the peripheral clock domain.	4	14

7.7.2 Software Watchdog Timer (SWT) configuration

SPC57EM80xx includes four SWTs.

Table 55. Software Watchdog Timer (SWT) instances

Instance	Description
SWT0	Intended for Main Core_0 in the high-speed computational clock domain.
SWT1	Intended for Main Core_1 in the high-speed computational clock domain.
SWT2	Intended for Peripheral Core_2 in the peripheral clock domain. Elapsed times recorded in timeout register are different from SWT0 and SWT1.
SWT3	Intended as security watchdog timer in the peripheral clock domain. Elapsed times recorded in timeout registers are different from SWT0 and SWT1

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Internal 16 MHz RC oscillator clock that is used for timer operation
- Programmable selection of system or internal oscillator clock for timer operation
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits

7.7.2.1 Security watchdog

SWT (SWT3 Security) is identical to the other SWT modules except the Instruction Address Compare feature is NOT instantiated. It is anticipated that each CPU would use an SWT to ensure software tasks continue to execute correctly.

If a CPU malfunctions (either software or hardware) the SWT is no longer serviced and a reset occurs. This is referred to as a Task Watchdog. The Security Watchdog is slightly different from the Task Watchdog.

The Security watchdog is intended to be serviced when the CPU executes an instruction at a specific address location. This address location would be an address within some fundamental section of security code. If this address location is not executed regularly, it implies that a malicious security attach has occurred and the device should be reset.

Additionally, the Security watchdog is deactivated by a device Power-On-Reset.

This security code could be executed on any of the CPUs. The SWT for that CPU would perform the Security Watchdog (SWT0, SWT1 or SWT2) because it includes the IAC feature. The Task Watchdog for that CPU would need to use SWT3 (Security), which does NOT include the IAC feature.

As a result the SWT3 (Security) is never used to perform the Security Watchdog; but is used for Task Watchdog to replace the SWT that IS being used for Security Watchdog

7.7.2.2 Reset assertion

The SWT can assert a reset when the watchdog timer expires. This reset causes a system reset equivalent to assertion of the RESET pin.

7.7.2.3 Default configuration

On SPC57EM80xx, the SWTs come out of reset with the following default values. [Table 56](#) summarizes the reset values. Minimum timeout value is 256 cycles for SWT0, SWT1, SWT2, and SWT3.

Table 56. Register reset values

SWT instance	Register	Reset value
SWT0	SWT Control Register (SWT_CR)	0xFF00_010A
SWT0	SWT Time-out (SWT_TO)	0x0005_FCD0
SWT1	SWT Control Register (SWT_CR)	0xFF00_010A
SWT1	SWT Time-out (SWT_TO)	0x0005_FCD0
SWT2	SWT Control Register (SWT_CR) ⁽¹⁾	0xFF00_011B
SWT2	SWT Time-out (SWT_TO)	0x0003_FDE0
SWT3	SWT Control Register (SWT_CR)	0xFF00_010A
SWT3	SWT Time-out (SWT_TO)	0x0005_FCD0

1. The SWT can be disabled by accepting a DCI-driven signal to clear the SWT_CR[WEN] bit. For SWT2, the default reset value for that bit is '1', but the DCI can override that default value by setting it to '0'. If the signal is de-asserted, there is no change to the SWT_CR[WEN] value. (The DCI can only disable the SWT, it cannot enable it.) See the DCI chapter for details on the system watchdog control.

The Security Watchdog (SWT3) is enabled using DCF Records in conjunction with the BAF. Please see [Section 58.3.6.2, Security watchdog configuration](#) for details.

For SWT[0,1,3], the SWT_CR register reset value of 0xFF00_011B selects:

- All masters allowed access
- Reset on invalid access
- Oscillator clock selected
- Counter stops in debug mode
- Watchdog is disabled

For SWT[2], the SWT_CR register reset value of 0xFF00_010A selects:

- All masters allowed access
- Reset on invalid access
- Oscillator clock selected
- Counter stops in debug mode
- Soft locked
- Watchdog is enabled

7.7.2.4 Service mode watchpoint input

Each SWT comes out of reset in 'Fixed Service Sequence' mode (SWT_CR[SMD] = 2'b00), but the Instruction Address Compare 8 (IAC8) register (SPR 568) can be used for servicing by using the SWT jd_watchpt input.

To use SWT address execution modes for this purpose, the e200z4 core IAC8 register signal jd_watchpt[15] is connected to the SWT jd_watchpt input. (See [Section 63.6, Watchpoint support](#) for details.) The jd_watchpt[15] signal tells the SWT that an IAC8 compare occurred.

Set SWT_CR[SMD] to one of the following so the SWT is serviced by executing code at the address loaded into the IAC8 register:

- 2'b10 for Fixed Address Execution; the IAC8 register cannot be updated while the watchdog is enabled. If the SWT is in 'Fixed Address Execution' mode, an SWT output signal tells the core; this 'freezes' changes to the IAC8 register when the SWT is enabled.
- 2'b11 for Incremental Address Execution; the IAC8 register can be updated.

7.7.3 PIT configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

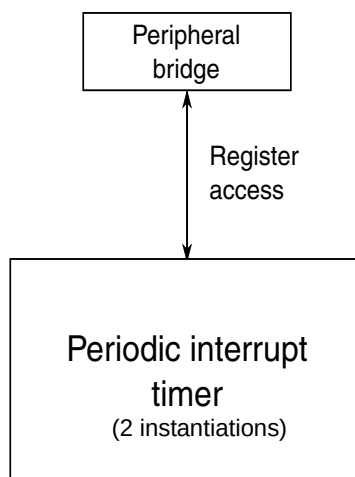


Figure 22. PIT configuration

Table 57. Reference links to related information

Topic	Related module	Reference
Full description	PIT	Chapter 45: Periodic Interrupt Timer (PIT)
System memory map	—	Chapter 5: Memory Map
Clocking	—	Chapter 27: Clocking
Power management	—	Chapter 10: Power management
DMA channel assignments	Direct memory access multiplexer	Chapter 26: DMACHMUX

7.7.3.0.1 PIT Instantiation

Table 58. PIT Information

PIT Module	No of Channels	Description
PIT0	8 + RTI	PIT0 has 8 standard channels and 1 RTI channel. In PIT0, all registers are cleared by any reset.
PIT1	2	PIT1 has chaining mode to implement a 64-bit timer and the timer value is unchanged by a functional reset.

7.7.3.0.2 PIT/DMA Periodic Trigger Assignments

In PIT1, DMA is not supported. The PIT0 generates periodic trigger events to the DMA Mux as shown in the table below.

Table 59. PIT0 channel assignments for periodic DMA triggering

DMA Channel	PIT0 Channel
DMA Channel 8	PIT0 Channel 0
DMA Channel 9	PIT0 Channel 1
DMA Channel 10	PIT0 Channel 2
DMA Channel 11	PIT0 Channel 3
DMA Channel 12	PIT0 Channel 4
DMA Channel 16	PIT0 Channel 5
DMA Channel 32	PIT0 Channel 6
DMA Channel 48	PIT0 Channel 7

7.7.3.0.3 PIT Registers

The following tables show the implementation of PIT registers in this MCU.

Table 60. PIT0 memory map

Absolute address (hex)	Register name
FFF8_4000	PIT Module Control Register (PIT0_MCR) ⁽¹⁾
FFF8_40E0 - FFF8_40E4	Reserved
FFF8_40F0 - FFF8_40FC	RTI Timer Channel Registers (PIT0_RTI_LDVAL, PIT0_RTI_CVAL, PIT0_RTI_TCTRL, PIT0_RTI_TFLG)
FFF8_4100 - FFF8_417C	Timer Channel [0:7] Registers (PIT0_LDVALn, PIT0_CVALn, PIT0_TCTRLn, PIT0_TFLGn)
FFF8_4180 - FFF8_7FFF	Reserved

1. The reset value of MCR is 0x06.

Table 61. PIT1 memory map

Absolute address (hex)	Register name
FFF8_0000	PIT Module Control Register (PIT1_MCR) ⁽¹⁾
FFF8_00E0	PIT Upper Lifetime Timer Register (PIT1_LTMR64H)
FFF8_00E4	PIT Lower Lifetime Timer Register (PIT1_LTMR64L)
FFF8_00F0 - FFF8_00F8	Reserved
FFF8_0100 - FFF8_011C	Timer Channel [0:1] Registers (PIT1_LDVALn, PIT1_CVALn, PIT1_TCTRLn, PIT1_TFLGn)
FFF8_0120 - FFF8_3FFF	Reserved

1. The reset value of MCR is 0x02.

7.7.3.1 PIT clocking

The PIT0 standard channels are clocked with the peripheral clock (PER_CLK).

The PIT0 RTI channel clock source is selectable between XOSC and RCOSC via the auxiliary clock selector 9.

The PIT 1 channels are clocked with the peripheral clock (PER_CLK).

7.7.4 GTM 103 Integration Module (GTMINT103) configuration

The GTM103 integration module (GTMINT) is an interface to the General Timer Module 103 (GTM103) that adds ECC functionality and a Nexus-based system debugger interface (GTMDI). It has two control registers and provides access to the GTM103's memory-mapped registers. The GTM103 is detailed separately in the *GTM-IP_103_Specification*. See [Chapter 88: GTM103 Integration \(GTMINT\) Module](#), for more information.

7.8 Communication interfaces

7.8.1 Fast Ethernet Controller (FEC) configuration

The Fast Ethernet Controller (FEC) is a communication controller supports 10 and 100 Mbps Ethernet/IEEE 802.3 networks.

7.8.2 FEC interface selection

On this chip, the FEC interface depends on the settings of the RCR (in the FEC) and SIUL2_SCR0 (in the SIUL2), as shown in [Table 62](#).

Table 62. FEC interface selection

RCR[MII_MODE]	SIUL2_SCR0[FEC_MODE]	Interface used by the FEC
0	0 or 1	7-wire
1	0	RMII
	1	MII ⁽¹⁾

1. MII supported only on BGA packages.

7.8.3 CAN subsystem configuration

7.8.3.1 CAN nodes

The number of CAN nodes are listed below:

Table 63. CAN bus controllers

Types	Instances	CAN FD support	Debug on CAN Support
M_CAN	M_CAN_1	Yes	Yes
	M_CAN_2	Yes	Yes
	M_CAN_3	Yes	No
M_CAN with TTCAN	M_TTCAN_0	Yes	No

7.8.4 LFAST and SIPI

This section summarizes the module configuration in the controller. For a comprehensive description of SIPI and LFAST, please reference the specific dedicated chapters for SIPI and LFAST.

7.8.4.1 Instantiation

The SIPI and LFAST are connected to one another and appear as a single unit. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules (see [Figure 23](#)).

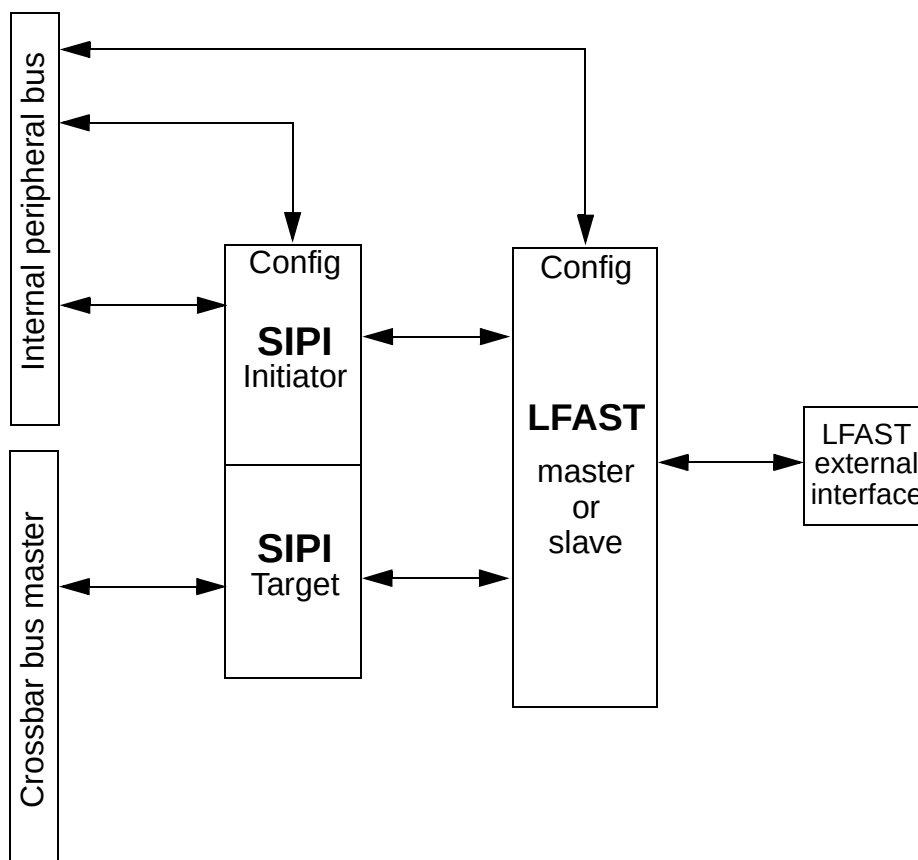


Figure 23. LFAST and SIPI block diagram

7.8.4.2 Device-specific features

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality. The SIPI CHIP ID = 0AF0A01Dh for the SPC57EM80xx.

7.8.4.3 Memory map and register description

Table 64. Reference text to related information

Topic	Related Module	Reference
Full description	LFAST	Chapter 49: LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications
Full description	SIPI	Chapter 47: Serial Interprocessor Interface (SIPI)

Table 64. Reference text to related information

Topic	Related Module	Reference
Direct memory access	eDMA	Chapter 25: Enhanced Direct Memory Access (eDMA)
System memory map	—	Chapter 5: Memory Map

7.8.5 DSPI configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

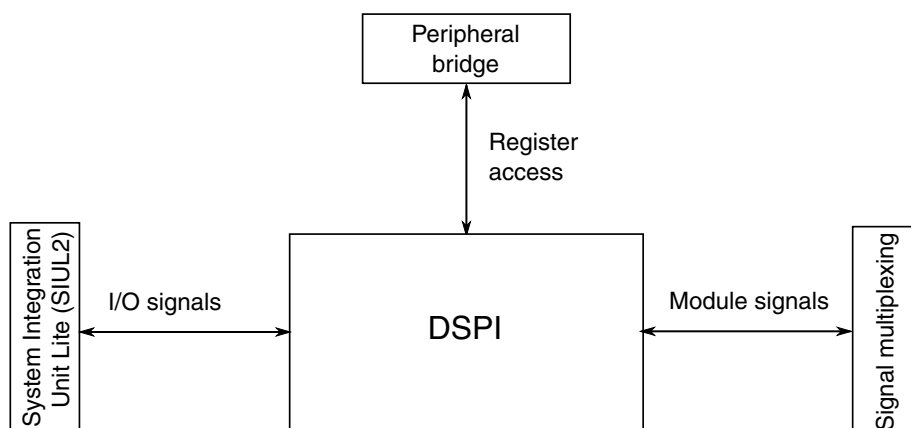


Figure 24. DSPI configuration

Table 65. Reference links to related information

Topic	Related module	Reference
Full description	DSPI	Chapter 48: Deserial Serial Peripheral Interface (DSPI)
System memory map		Chapter 5: Memory Map
Clocking		Chapter 27: Clocking
Signal Multiplexing	Port control	Chapter 16: System Integration Unit Lite2 (SIUL2)
System Module	System Integration Unit Lite	Chapter 16: System Integration Unit Lite2 (SIUL2)
DMA Channel Assignments	Direct memory access multiplexer	Chapter 26: DMACHMUX

7.8.5.1 DSPI instantiation

This device contains seven DSPI modules. [Table 66](#) shows the instantiation information for each module.

Table 66. DSPI instantiation

DSPI Module	CTARs	TX FIFO Depth	RX FIFO Depth	PCS Signals	LVDS Support	Data Serialization Support
DSPI0	8	4	4	PCS[7:0]	Not Available	—
DSPI1	8	4	4	PCS[7:0]	Not Available	—
DSPI2	8	4	4	PCS[7], PCS[4:0]	Available ⁽¹⁾	—
DSPI3	8	4	4	PCS[7:0]	Not Available	—
DSPI4	8	4	4	PCS[7:0]	Available ⁽²⁾	32
DSPI5	8	4	4	PCS[7:0]	Available ⁽³⁾	32
DSPI12 ⁽⁴⁾	8	4	4	—	Not Available	—

1. Does not support Microsecond Bus.
2. Supports Microsecond Bus SCK and SOUT LVDS signal pairs.
3. Supports Microsecond Bus SCK and SOUT LVDS signal pairs; supports Differential DSPI with SCK, SIN and SOUT LVDS signal pairs.
4. Available in half duplex mode.

7.8.5.2 DSPI half duplex operation

Half duplex operation is not supported within the DSPI module, but it is configured by connecting the SIN and SOUT together within the SIUL2 and operating the combined pad in open drain mode. Software must ensure that the SOUT signal is passive (high) during SIN reception. This can be done by reconfiguring the SIUL2 to disconnect the SOUT or by writing a data frame that is all '1's.

7.8.5.3 Serialized timed I/O

The timer sources and injection pin inputs to be DSPI serialized downstream outputs are selected through the SIUL2 and selected through the MSCR registers. Inversion of these signals is also handled in the SIUL2. Refer to [Chapter 16: System Integration Unit Lite2 \(SIUL2\)](#), for details.

7.8.5.4 High Speed and MicroSecond Channel (MSC) Support

The baud rate generator in the DSPI is clocked from the peripheral clock. The DSPI module can operate at a baud rate of up to half the peripheral clock. The peripheral clock frequency specification is 16 MHz to a maximum of 100 MHz. As long as the peripheral clock is 100 MHz, 50 MHz baud rate is achievable.

To support microsecond channel functionality, pairs of LVDS signals are implemented for the appropriate DSPI signals. Therefore, SOUT4, SOUT_5, SOUT_6, SCK_4, SCK_5, SCK_6, have LDVS capability. In addition, SIUL2 MSCR muxing is implemented to mux the SOUT and CLK onto these LVDS signals well as standard CMOS signals. Finally, in the SIUL2 MSCR pin muxing, 4 input pins can be routed to 32 possible serialization inputs, each selectable for a specific DSPI.

The following four external pins can be inserted into the MSC data stream:

- 0000_0010 IO_PAD PC[2]
- 0000_0011 IO_PAD PD[8]
- 0000_0100 IO_PAD PF[3]
- 0000_0101 IO_PAD PJ[4]

Details of these pin assignments can be found in the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the Input Multiplexing Table tab.

7.8.5.5 DSPI HT external trigger

External hardware triggers are not supported.

7.8.5.6 DSPI ITSB mode

For the ITSB trigger, the internal DSPI trigger (TRGPRD in DSICR1) is used. Use of external trigger is not supported.

7.8.5.7 DSPI registers

The following table shows the absolute addresses of DSPI module registers in this MCU.

Table 67. DSPI registers memory map

Absolute address (hex)	Module
FBE7_0000	DSPI2 registers
FBE7_4000	DSPI3 registers
FBE7_8000	DSPI5 registers
FFE7_0000	DSPI0 registers
FFE7_4000	DSPI1 registers
FFE7_8000	DSPI4 registers
FFE8_8000	DSPI12 registers

7.8.5.7.1 DSPIx_MCR implementation

The following figure shows the implementation of DSPIx_MCR register in this MCU.

Figure 25. DSPI Module Configuration Register (DSPIx_MCR)

Address : Base + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSTR	CONT_SCKE	DCONF		FRZ	MTFE	PCSSE	ROOE	PCISIS[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DOZE ⁽¹⁾	MDIS	DIS_TXF	DIS_RXF	0	0	SMPL_PT		0	0	0	0	XSPI	FCPCS	PES	HALT
W					CLR_TXF	CLR_RXF										
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Doze mode is not implemented on SPC57EM80xx.

7.8.6 FlexRay Configuration

The FlexRay communications system is designed to provide high-speed deterministic distributed control for advanced automotive applications. Its dual-channel architecture supporting data rates up to 10 Mbit/s per channel offers system-wide redundancy that supports the reliability requirements of safety system.

7.8.6.1 FlexRay signals pin assignment

This table lists the pin names for the FlexRay signals that are connected to pins on the boundary of the chip.

Table 68. Flexray pin names

Signal description	Pin name
FR_A_RX (Receive Data Channel A)	RXDA
FR_A_TX (Transmit Data Channel A)	CA_TX
FR_A_TX_EN (Transmit Enable Channel A)	CA_TR_EN
FR_B_RX (Receive Data Channel B)	RXDB
FR_B_TX (Transmit Data Channel B)	CB_TX
FR_B_TX_EN (Transmit Enable Channel B)	CB_TR_EN
FR_DBG[0] (Debug Strobe Signal 0)	DBG[0]
FR_DBG[1] (Debug Strobe Signal 1)	DBG[1]
FR_DBG[2] (Debug Strobe Signal 2)	DBG[2]
FR_DBG[3] (Debug Strobe Signal 3)	DBG[3]

7.8.6.2 Stopwatch timer input

Stopwatch timer input is routed to FlexRay module (selected by SIUL MSCR register) via:

- STM0_CMP0
- STM1_CMP0
- STM2_CMP0
- PIT1_CH0

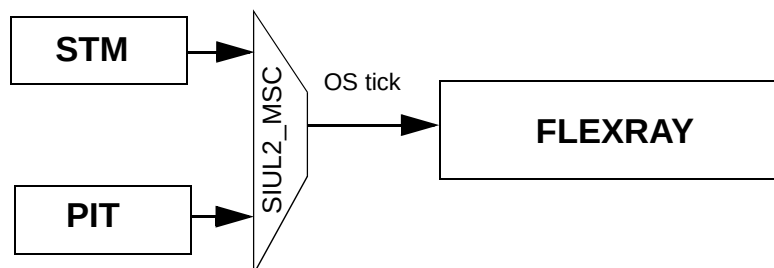


Figure 26. Stopwatch timer source muxing

7.8.7 PSI5 Controller Configuration

The PSI5 chapter has 2 instances. Parameters for the 2 instances are listed below:

Table 69. PSI5 controller generics.

Signal name	Description	Value - PSI5_0	Value - PSI5_1
PSI5_NUM_CH	Number of channels	2	1
PSI5_PDIS_RST	Default State of the IP after reset	1	1

7.8.8 SENT receiver (SRX) configuration

7.8.8.1 SRX channels

The following table shows the SENT receiver channels for each instance configured on this chip. SPC57EM80xx has two SENT modules.

7.8.8.2 SRX channels

The following table shows the SENT receiver channels for each instance configured on this chip.

Table 70. SENT channel configuration

Instance	Channels (CH)
SENT 0	5 Channels: SENT_0_CH0 SENT_0_CH1 SENT_0_CH2 SENT_0_CH3 SENT_0_CH4
SENT 1	5 Channels: SENT_1_CH0 SENT_1_CH1 SENT_1_CH2 SENT_1_CH3 SENT_1_CH4

7.8.8.3 SRX registers

The following table lists the address offset and memory map for both SENT receiver (SRX) instances for this chip.

Table 71. SENT Receiver memory map

Offset	Register Name
0x0000	GBL_CTRL—Global Control Register
0x0004	CHNL_EN—Channel Enable Register
0x0008	GBL_STATUS—Global Status Register
0x000C	FMSG_RDY—Fast Message Ready Register
0x0010	SMSG_RDY—Slow Serial Message Ready Register
0x0014	Reserved
0x0018	DATA_CTRL1—Data Control Register 1
0x001C	DATA_CTRL2—Data Control Register 2
0x0020	Reserved
0x0024	Reserved
0x0028	FDMA_CTRL—Fast Message DMA Control Register
0x002C	SDMA_CTRL—Slow Serial Message DMA Control Register
0x0030	Reserved
0x0034	FRDY_IE—Fast Message Ready Interrupt Control Register
0x0038	SRDY_IE—Slow Serial Message Ready Interrupt Control Register
0x003C	Reserved
0x0040	DMA_FMSG_DATA—DMA Fast Message Read Register
0x0044	DMA_FMSG_CRC—DMA Fast Message CRC Read Register
0x0048	DMA_FMSG_TS—DMA Fast Message Time Stamp Read Register

Table 71. SENT Receiver memory map (continued)

Offset	Register Name
0x004C	Reserved
0x0050	DMA_SMSG_BIT3—DMA Serial Message Read Register (Bit 3)
0x0054	DMA_SMSG_BIT2—DMA Serial Message Read Register (Bit 2)
0x0058	DMA_SMSG_TS—DMA Serial Message Time Stamp Read Register
0x005C	Reserved
0x0060	CH0_CLK_CTRL—Channel 'n' Clock Control Register
0x0064	CH0_STATUS—Channel 'n' Status Register
0x0068	CH0_CONFIG—Channel 'n' Configuration Register
0x006C	Reserved
0x0070	CH1_CLK_CTRL—Channel 'n' Clock Control Register
0x0074	CH1_STATUS—Channel 'n' Status Register
0x0078	CH1_CONFIG—Channel 'n' Configuration Register
0x007C	Reserved
0x0080	CH2_CLK_CTRL—Channel 'n' Clock Control Register
0x0084	CH2_STATUS—Channel 'n' Status Register
0x0088	CH2_CONFIG—Channel 'n' Configuration Register
0x008C	Reserved
0x0090	CH3_CLK_CTRL—Channel 'n' Clock Control Register
0x0094	CH3_STATUS—Channel 'n' Status Register
0x0098	CH3_CONFIG—Channel 'n' Configuration Register
0x009C	Reserved
0x00A0	CH4_CLK_CTRL—Channel 'n' Clock Control Register
0x00A4	CH4_STATUS—Channel 'n' Status Register
0x00A8	CH4_CONFIG—Channel 'n' Configuration Register
0x00AC - 0x015F	Reserved
0x0160	CH0_FMSG_DATA—Channel 'n' Fast Message Register
0x0164	CH0_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x0168	CH0_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x016C	CH0_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x0170	CH0_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x0174	CH0_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x0178	CH1_FMSG_DATA—Channel 'n' Fast Message Register
0x017C	CH1_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x0180	CH1_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x0184	CH1_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)

Table 71. SENT Receiver memory map (continued)

Offset	Register Name
0x0188	CH1_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x018C	CH1_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x0190	CH2_FMSG_DATA—Channel 'n' Fast Message Register
0x0194	CH2_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x0198	CH2_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x019C	CH2_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x01A0	CH2_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x01A4	CH2_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x01A8	CH3_FMSG_DATA—Channel 'n' Fast Message Register
0x01AC	CH3_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x01B0	CH3_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x01B4	CH3_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x01B8	CH3_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x01BC	CH3_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x01C0	CH4_FMSG_DATA—Channel 'n' Fast Message Register
0x01C4	CH4_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x01C8	CH4_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x01CC	CH4_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x01D0	CH4_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x01D4	CH4_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x01D8 - 0x02E0	Reserved

7.8.9 LINFlexD — Configurations

The LINFlexD chapter has 5 instances. Configuration for the 5 instances are listed below:

Table 72. LINFlexD configurations

Description	linflex_0	linflex_1	linflex_2	linflex_14	linflex_15
Number of filters implemented	16	16	16	16	16
Number of Tx DMA channels	1	1	1	1	1
Number of Rx DMA channels	1	1	1	1	1
LIN operation mode	master/slave	master	master	master	master
Autosynchronization	yes	no	no	no	no

7.9 Reset and boot modules

7.9.1 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration in the SPC57EM80xx. For a comprehensive description of the SSCM, please reference the SSCM's dedicated chapter.

7.9.1.1 SSCM instantiation

There is one SSCM instance in the SPC57EM80xx device. The SSCM is part of the reset and boot sub-system.

7.9.1.2 Device specific features

The SSCM allows Boot Assist from Flash (BAF), LINFlexD search at boot and utilizes PASS security concept. It also allows for external booting from the LINFlex port, but with no autobaud feature. The device specific resets value for the SSCM_MEMCONFIG register are shown in [Table 73](#).

Table 73. SSCM_MEMCONFIG Reset Values

Field	Field Definition	Value
MREV	Minor Revision	0000b ⁽¹⁾
JPIN	JTAG Part ID	30Ah ⁽²⁾

1. This value will change for each minor revision of the device.
2. This value will change for each major revision of the device.

[Table 74](#) shows the address locations that the SSCM uses to search for startup information.

Table 74. SSCM startup information addresses

Description	Value
The location at which the SSCM will look for the UTEST DCF records (UTEST_OFFSET).	0040_0000h
Location of the Boot Assist code in Flash (BAF)	0040_4000h
Specify the locations which are searched for a valid HSM boot header. (starting from the lowest address)	60_C000h 61_0000h 62_0000h

7.9.1.3 Protected registers

The SSCM module has a built-in register protection mechanism that affects the behavior of write operations to one or more registers. After software has activated the protection for a register, writes can only be performed in supervisor mode until the protection is either deactivated by software or the device undergoes reset.

See the Register protection section of this chapter for a list of affected registers.

See the Register protection chapter for a detailed description of the register protection mechanism.

7.9.1.4 Memory map and register description

Table 75. Reference text to related information

Topic	Related Module	Reference
Reset	Reset Generation Module (MC_RGM)	
Reset/Boot	Reset and Boot	Chapter 8: Reset and Boot
Boot	Boot Assist from Flash (BAF)	Chapter 58: Boot Assist Flash (BAF)
Register Protection	Register Protection (REG_PROT)	Chapter 83: Register Protection (REG_PROT)
Register Protection	Register Protection Configuration	Section 7.10.5.12, System Status and Control Module (SSCM) protected registers
Serial Boot	LINFlexD	Chapter 56: LINFlexD
Security	PASS	Chapter 84: Password and Device Security Module (PASS)
Security	HSM	Chapter 11: Security

7.10 Safety modules

7.10.1 CRC – Number Of Contexts

The number of contexts available for the current CRC computation of multiple data streams of CRC are listed below:

Table 76. CRC Contexts

Signal name	Description	Value
CRC_CNTX_NUM (N)	Number of contexts available for the current CRC computation of multiple data streams	1

7.10.2 MEMU configuration

7.10.2.1 MEMU system connections

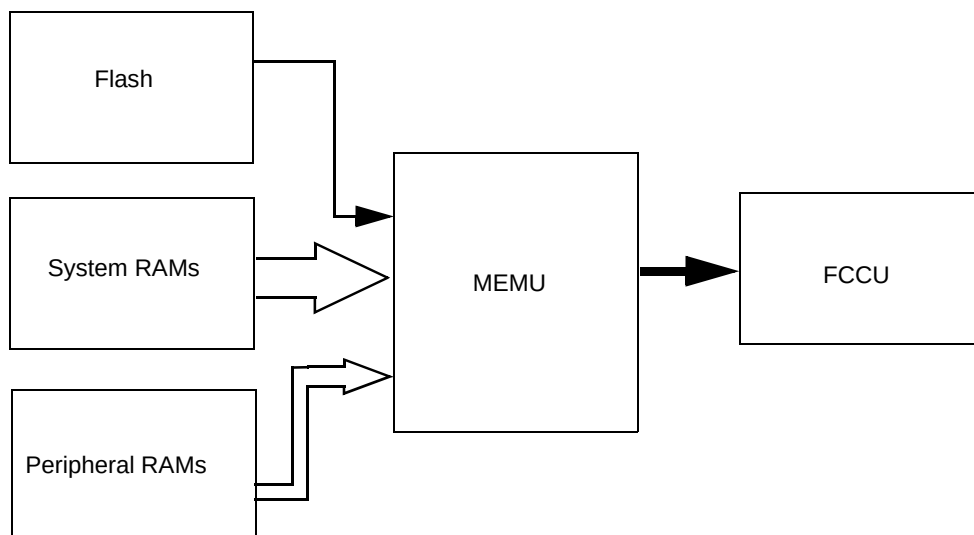


Figure 27. MEMU system connections

7.10.2.2 MEMU error sources

The following table shows the MEMU error sources for each instance configured on this chip.

Table 77. MEMU error sources

Instance	Value
Number of System RAM unique error sources	93
Number of Peripheral RAM unique error sources	25
Number of FLASH unique error sources	3

7.10.2.3 Reporting tables implementation

There are 3 categories of error reporting for the MEMU:

- Flash error reporting
- System RAM error reporting—Any RAM that is CPU accessible falls into this category.
- Peripheral RAM error reporting—Any RAM that is not CPU accessible falls into this category.

Each category of error reporting has two tables associated with it, these are:

- Correctable Error Reporting Table
- Uncorrectable Error Reporting Table

Each entry in a reporting table corresponds to a unique error event. The number of entries supported by each table depends upon the error source.

The table below describes the number of entries in each reporting table for the different error sources.

Table 78. Number of entries in each reporting table

Error Source	Number of entries in correctable error reporting table	Number of entries in un-correctable error reporting table
Flash	20	1
System RAM	10	1
Peripheral RAM	2	1

7.10.2.4 Concurrent overflow register (OFLW) implementation

Errors are reported to the MEMU from one of three categories:

- Flash
- System RAM
- Peripheral RAM

In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLW register details the corresponding error source to the respective bit-field in the Concurrent Overflow registers.

Table 79. Corresponding error sources for System RAM OFLW0 register

OFLW0 bit-field position (bit)	Error sources
0	z4a Data AHB
1	z4a Instruction AHB
2	z4a Data TCM (DMEM)
3	z4a Instruction TCM (IMEM)
4	z4a Data Cache
5	z4a Instruction Cache
6	z4b Data AHB
7	z4b Instruction AHB
8	z4b Data TCM (DMEM)
9	z4b Instruction TCM (IMEM)
10	z4b Data Cache
11	z4b Instruction Cache
12	z4c Instruction Cache
13	z4c Data AHB
14	z4c Instruction AHB
15	z4c Data TCM (DMEM)
16	z4c Instruction TCM (IMEM)

Table 79. Corresponding error sources for System RAM OFLW0 register

OFLW0 bit-field position (bit)	Error sources
17	SRAM Memory
18	PBRIDGE0
19	MBIST z4a Instruction TCM
20	MBIST z4a Data TCM Instance 1
21	MBIST z4a Data TCM Instance 2
22	MBIST z4a Instruction Cache Instance 1
23	MBIST z4a Instruction Cache Instance 2
24	MBIST z4a Instruction Cache Instance 3
25	MBIST z4a Instruction Cache Instance 4
26	MBIST z4a Data Cache Instance 1
27	MBIST z4a Data Cache Instance 2
28	MBIST z4a Data Cache Instance 3
29	MBIST z4a Data Cache Instance 4
30	MBIST z4a Instruction Cache Tag
31	MBIST z4a Data Cache Tag

Table 80. Corresponding error sources for System RAM OFLW1 register

OFLW1 bit-field position (bit)	Error Sources
0	MBIST z4b Instruction TCM
1	MBIST z4b Data TCM Instance 1
2	MBIST z4b Data TCM Instance 2
3	MBIST z4b Instruction Cache Instance 1
4	MBIST z4b Instruction Cache Instance 2
5	MBIST z4b Instruction Cache Instance 3
6	MBIST z4b Instruction Cache Instance 4
7	MBIST z4b Data Cache Instance 1
8	MBIST z4b Data Cache Instance 2
9	MBIST z4b Data Cache Instance 3
10	MBIST z4b Data Cache Instance 4
11	MBIST z4b Instruction Cache Tag
12	MBIST z4b Data Cache Tag
13	MBIST z4c Instruction Memory
14	MBIST z4c Data Memory Instance 1
15	MBIST z4c Data Memory Instance 2
16	MBIST z4c Instruction Cache Instance 1

Table 80. Corresponding error sources for System RAM OFLW1 register

OFLW1 bit-field position (bit)	Error Sources
17	MBIST z4c Instruction Cache Instance 2
18	MBIST z4c Instruction Cache Instance 3
19	MBIST z4c Instruction Cache Instance 4
20	MBIST z4c Instruction Cache Tag
21	MBIST RAM 0
22	MBIST RAM 1 Instance 1
23	MBIST RAM 1 Instance 2
24	MBIST Instruction Cache Instance 1
25	MBIST Instruction Cache Instance 2
26	MBIST Instruction Cache Tag Instance 1
27	MBIST Instruction Cache Tag Instance 2
28	MBIST HSM Crypto Channel Controller (C3)
29	MBIST DMA
30	MBIST FlexRay Data RAM
31	MBIST FlexRay Instruction RAM

Table 81. Corresponding error sources for System RAM OFLW2 register

OFLW2 bit-field position (bit)	Error Sources
0	MBIST Controller Area Network
1	MBIST GTM FIFO
2	MBIST GTM RAM0 Instance 1
3	MBIST GTM RAM0 Instance 2
4	MBIST GTM RAM0 Instance 3
5	MBIST GTM RAM0 Instance 4
6	MBIST GTM RAM1 Instance 1
7	MBIST GTM RAM1 Instance 2
8	MBIST GTM RAM1 Instance 3
9	MBIST GTM RAM1 Instance 4
10	MBIST Digital PLL Instance 1a
11	MBIST Digital PLL Instance 1b
12	MBIST Digital PLL Instance 2
13	MBIST SRAM Instance 1
14	MBIST SRAM Instance 2
15	MBIST Nexus Instance 1
16	MBIST Nexus Instance 2

Table 81. Corresponding error sources for System RAM OFLW2 register

OFLW2 bit-field position (bit)	Error Sources
17	MBIST Nexus Instance 3
18	MBIST Nexus Instance 4
19	MBIST AMU
20	MBIST FEC FIFO
21	MBIST FEC MIB
22	MBIST BAM
23	MBIST Overlay Instance 0
24	MBIST Overlay Instance 1
25	Decorated PRIDGE1
26	Decorated PRAM
27	Decorated z4a TCM
28	Decorated z4b TCM
29	Decorated z4c TCM
30	Reserved
31	Reserved

Table 82. Corresponding error sources for Peripheral RAM OFLW register

OFLW bit-field position (bit)	Error Sources
0	DMA
1	Advanced Memory Unit
2	Concentrator 0
3	Concentrator 1
4	AIPS Bridge 0
5	AIPS Bridge 1
6	FEC RIF RAM
7	FEC MIB RAM
8	HSM Platform RAM
9	HSM Instruction Cache RAM
10	FlexRay Data RAM
11	FlexRay LUT RAM
12	Controller Area Network
13	GTM FIFO
14	GTM Digital PLL r1a
15	GTM Digital PLL r1b

Table 82. Corresponding error sources for Peripheral RAM OFLW register

OFLW bit-field position (bit)	Error Sources
16	GTM Digital PLL r2
17	GTM Multi Channel Sequencer0 core0
18	GTM Multi Channel Sequencer1 core0
19	GTM Multi Channel Sequencer0 core1
20	GTM Multi Channel Sequencer1 core1
21	GTM Multi Channel Sequencer0 core2
22	GTM Multi Channel Sequencer1 core2
23	GTM Multi Channel Sequencer0 core3
24	GTM Multi Channel Sequencer1 core3
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

Table 83. Corresponding error sources for FLASH OFLW register

OFLW0 bit-field position (bit)	Error sources
0	Flash 0
1	Flash 1
3	PFLASH AMU
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved

Table 83. Corresponding error sources for FLASH OFLW register

OFLW0 bit-field position (bit)	Error sources
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

7.10.2.5 MEMU registers

The following table shows the address offsets and implementation of the MEMU registers in this chip.

Table 84. MEMU registers

Offset	Register
0x0000	Control register (CTRL)
0x0004	Error flag register (ERR_FLAG)
0x000C	Debug register (DEBUG)
0x0020	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS0)
0x0024	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR0)
0x0028	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS1)
0x002C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR1)
0x0030	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS2)
0x0034	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR2)
0x0038	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS3)
0x003C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR3)
0x0040	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS4)
0x0044	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR4)

Table 84. MEMU registers (continued)

Offset	Register
0x0048	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS5)
0x004C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR5)
0x0050	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS6)
0x0054	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR6)
0x0058	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS7)
0x005C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR7)
0x0060	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS8)
0x0064	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR8)
0x0068	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS9)
0x006C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR9)
0x0070	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)
0x0074	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)
0x0078	System RAM concurrent overflow register (SYS_RAM_OFLW0)
0x007C	System RAM concurrent overflow register (SYS_RAM_OFLW1)
0x0080	System RAM concurrent overflow register (SYS_RAM_OFLW2)
0x0620	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS0)
0x0624	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR0)
0x0628	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS1)
0x062C	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR1)
0x0630	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)
0x0634	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)
0x0638	Peripheral RAM concurrent overflow register (PERIPH_RAM_OFLW0)
0x0C20	Flash memory correctable error reporting table status register (FLASH_CERR_STS0)
0x0C24	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR0)
0x0C28	Flash memory correctable error reporting table status register (FLASH_CERR_STS1)
0x0C2C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR1)
0x0C30	Flash memory correctable error reporting table status register (FLASH_CERR_STS2)
0x0C34	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR2)
0x0C38	Flash memory correctable error reporting table status register (FLASH_CERR_STS3)
0x0C3C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR3)
0x0C40	Flash memory correctable error reporting table status register (FLASH_CERR_STS4)
0x0C44	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR4)
0x0C48	Flash memory correctable error reporting table status register (FLASH_CERR_STS5)
0x0C4C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR5)

Table 84. MEMU registers (continued)

Offset	Register
0x0C50	Flash memory correctable error reporting table status register (FLASH_CERR_STS6)
0x0C54	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR6)
0x0C58	Flash memory correctable error reporting table status register (FLASH_CERR_STS7)
0x0C5C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR7)
0x0C60	Flash memory correctable error reporting table status register (FLASH_CERR_STS8)
0x0C64	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR8)
0x0C68	Flash memory correctable error reporting table status register (FLASH_CERR_STS9)
0x0C6C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR9)
0x0C70	Flash memory correctable error reporting table status register (FLASH_CERR_STS10)
0x0C74	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR10)
0x0C78	Flash memory correctable error reporting table status register (FLASH_CERR_STS11)
0x0C7C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR11)
0x0C80	Flash memory correctable error reporting table status register (FLASH_CERR_STS12)
0x0C84	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR12)
0x0C88	Flash memory correctable error reporting table status register (FLASH_CERR_STS13)
0x0C8C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR13)
0x0C90	Flash memory correctable error reporting table status register (FLASH_CERR_STS14)
0x0C94	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR14)
0x0C98	Flash memory correctable error reporting table status register (FLASH_CERR_STS15)
0x0C9C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR15)
0x0CA0	Flash memory correctable error reporting table status register (FLASH_CERR_STS16)
0x0CA4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR16)
0x0CA8	Flash memory correctable error reporting table status register (FLASH_CERR_STS17)
0x0CAC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR17)
0x0CB0	Flash memory correctable error reporting table status register (FLASH_CERR_STS18)
0x0CB4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR18)
0x0CB8	Flash memory correctable error reporting table status register (FLASH_CERR_STS19)
0x0CBC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR19)
0x0CC0	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)
0x0CC4	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)
0x0CC8	Flash memory concurrent overflow register (FLASH_OFLW0)

7.10.3 Indirect Memory Access (IMA) configuration

The IMA refers to the alteration of memory data values during system development and test activities. This section provides details about what memory locations can be accessed—

either via processor core access or via the IMA module—and how they are configured. For details on IMA see [Chapter 80: Indirect Memory Access \(IMA\)](#).

Note: See the *SPC57EM80xx Microcontroller Security Reference Manual for the HSM IMA details*.

7.10.3.1 SRAM Arrays accessible by processor cores

All the SRAM arrays in [Table 85](#) can be accessed by the main CPUs (CPU0 & CPU1) and the IOP (CPU2).

Table 85. SRAM with processor Core access

Block	Function	SPC57EM80xx Cut1		
		Words	Bits	ECC Bits
Safety Core	IMEM	2048	72	—
	DMEM	8192	40	—
		8192	40	—
	Icache	256	72	—
		256	72	—
		256	72	—
		256	72	—
	Dcache	128	80	—
		128	80	—
		128	80	—
		128	80	—
	ITAG	128	65	—
	DTAG	64	71	—
Comp Core	IMEM	2048	72	—
	DMEM	8192	40	—
		8192	40	—
	Icache	256	72	—
		256	72	—
		256	72	—
		256	72	—
	Dcache	128	80	—
		128	80	—
		128	80	—
		128	80	—
	ITAG	128	65	—
	DTAG	64	71	—

Table 85. SRAM with processor Core access (continued)

Block	Function	SPC57EM80xx Cut1		
		Words	Bits	ECC Bits
IOP Core	IMEM	2048	72	—
	DMEM	4096	40	—
		4096	40	—
	Icache	256	72	—
		256	72	—
		256	72	—
		256	72	—
	ITAG	128	65	—
System RAM	System RAM	8192	72	—
		8192	72	—
		—	—	—
		—	—	—
		—	—	—
		—	—	—
Overlay	Overlay	1024	72	—
		1024	72	—

7.10.3.2 Memory accessible via the IMA module

The SRAM arrays in [Table 86](#) are accessible via the IMA Module.

Table 86. SRAM arrays accessible via IMA

IMA ARRAY SELECT	Block	Function	SPC57EM80xx Cut1		
			Words	Bits	ECC Bits
0	None	IMA Disabled	—	—	—
1	DMA	DMA	256	72	71–64
2			—	—	—
3	FlexRay	DRAM	128	26	25–16
4			—	—	—
5		LRAM	96	126	125–96
6			—	—	—
7			—	—	—
8			—	—	—

Table 86. SRAM arrays accessible via IMA (continued)

IMA ARRAY SELECT	Block	Function	SPC57EM80xx Cut1		
			Words	Bits	ECC Bits
9	TTCAN	TTCAN	3968	39	38–32
10	GTM	FIFO	1024	36	35–29
11			—	—	—
12			—	—	—
13		MCS RAM0	1024	39	38–32
14			1024	39	38–32
15			1024	39	38–32
16			1024	39	38–32
17			—	—	—
18			—	—	—
19		MCS RAM1	512	39	38–32
20			512	39	38–32
21			512	39	38–32
22			512	39	38–32
23			—	—	—
24			—	—	—
25		DPLL RAM1a	96	31	30–24
26		DPLL RAM1b	384	31	30–24
27		DPLL RAM2	2048	31	30–24
28	AMU	AMU	3200	144	143–136, 71–64
29	FEC	FIFO (RIF)	128	44	43–36
30		MIB	64	40	39–32

7.10.3.3 Memory with inaccessible ECC bits

The following table shows memory locations that cannot be accessed.

Table 87. No ECC bit access

IMA ARRAY SELECT	Block	Function	SPC57EM80xx Cut1		
			Words	Bits	ECC Bits
No IMA Access Required - Debug function					

Table 87. No ECC bit access (continued)

IMA ARRAY SELECT	Block	Function	SPC57EM80xx Cut1		
			Words	Bits	ECC Bits
—	NAR	NAR	64	128	—
—			64	128	—
—			64	128	—
—			64	128	—

7.10.4 FCCU configuration

Note: See [Section 6.4.1.7.3, FCCU failure inputs from other modules](#) for a list FCCU failure inputs

7.10.4.1 FCCU register availability and reset values

[Table 88](#) shows the availability and reset values of selected FCCU registers.

Table 88. Availability and reset values of selected FCCU registers

Register	Reset value
FCCU_RF_CFG0	0000_0000_0000_0000b
FCCU_RF_CFG1	0000_0000_0000_0000b
FCCU_RF_CFG2	0000_0000_0000_0000b
FCCU_RF_CFG3	0000_0000_0000_0000b
FCCU_RFS_CFG0	0000_0000_0000_0000b
FCCU_RFS_CFG1	0000_0000_0000_0000b
FCCU_RFS_CFG2	0000_0000_0000_0000b
FCCU_RFS_CFG3	0000_0000_0000_0000b
FCCU_RFS_CFG4	0000_0000_0000_0000b
FCCU_RFS_CFG5	0000_0000_0000_0000b
FCCU_RFS_CFG6	0000_0000_0000_0000b
FCCU_RFS_CFG7	0000_0000_0000_0000b
FCCU_RFE0	1111_1111_1111_1111b
FCCU_RFE1	1111_1111_1111_1111b
FCCU_RFE2	1111_1111_1111_1111b
FCCU_RFE3	1111_1111_1111_1111b
FCCU_RF_TOE0	1111_1111_1111_1111b
FCCU_RF_TOE1	1111_1111_1111_1111b
FCCU_RF_TOE2	1111_1111_1111_1111b
FCCU_RF_TOE3	1111_1111_1111_1111b

7.10.4.2 FOSU parameters

The FOSU_COUNT, or FOSU time out response time, is 1000 IRCOSC clock cycles.

7.10.5 Register protection (REG_PROT) configuration

Some modules on this device include a built-in register protection mechanism that can be used to override the normal write permissions associated with individual registers in the module. This mechanism is software-controlled and can be dynamically reconfigured at runtime or made effective until device reset. For a detailed explanation, see the [Chapter 83: Register Protection \(REG_PROT\)](#).

Table 89. Reference links to related information

Topic	Related module	Reference
System memory map	—	Chapter 5: Memory Map
SIUL2 registers	SIUL2	Chapter 16: System Integration Unit Lite2 (SIUL2)
CMU registers	CMU, CMUIOP	Chapter 29: Clock Monitor Unit (CMU)
PMC Digital Interface (PMC_DIG) registers	PMC_DIG	Chapter 60: Power Management Controller digital interface (PMC_dig)
PLL Digital Interface (PLL_DIG) registers	PLL_DIG	Chapter 28: Dual PLL Digital Interface (PLLDIG)
Oscillator Digital Interface (OSC_DIG) registers	OSC_DIG	Chapter 31: OSC Digital Interface (XOSC)
Memory Error Management Unit (MEMU) registers	MEMU	Chapter 79: Memory Error Management Unit (MEMU)
Mode Entry (MC_ME) registers	MC_ME	Chapter 62: Mode Entry Module (MC_ME)
Clock Generation Module (MC_CGM) registers	MC_CGM	Chapter 30: Clock Generation Module (MC_CGM)
Reset Generation Module (MC_RGM) registers	MC_RGM	Chapter 57: Reset Generation Module (MC_RGM)
System Status and Control Module (SSCM) registers	SSCM	Chapter 59: System Status and Configuration Module (SSCM)
Indirect Memory Access (IMA) registers	IMA	
JTAG Master (JTAGM) registers	JTAGM	

7.10.5.1 SIUL2 protected registers

Table 90 lists the SIUL2 registers that can be protected.

Table 90. Protected SIUL2 registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SIUL_DIRER0	32	0x0018	32
SIUL_DIRSR0	32	0x0020	32
SIUL_IREEER0	32	0x0028	32
SIUL_IFEER0	32	0x0030	32
MSCR 0-15	32	0x0240	32 (x16)
MSCR 24-27	32	— ⁽¹⁾	32 (x4)
MSCR 32-47	32	— ⁽¹⁾	32 (x16)
MSCR 49	32	— ⁽¹⁾	32
MSCR 51-54	32	— ⁽¹⁾	32 (x4)
MSCR 56-76	32	— ⁽¹⁾	32 (x21)
MSCR 80-89	32	— ⁽¹⁾	32 (x10)
MSCR 91	32	— ⁽¹⁾	32
MSCR 109	32	— ⁽¹⁾	32
MSCR 110-127	32	— ⁽¹⁾	32 (x18)
MSCR 136-148	32	— ⁽¹⁾	32 (x13)
MSCR 149-159	32	— ⁽¹⁾	32 (x11)
MSCR 172-202	32	— ⁽¹⁾	32 (x31)
MSCR 512-519	32	— ⁽¹⁾	32 (x8)
MSCR 520-527	32	— ⁽¹⁾	32 (x8)
MSCR 528-535	32	— ⁽¹⁾	32 (x8)
MSCR 536-543	32	— ⁽¹⁾	32 (x8)
MSCR 576-577	32	— ⁽¹⁾	32 (x12)
MSCR 578-579	32	— ⁽¹⁾	32 (x2)
MSCR 580-581	32	— ⁽¹⁾	32 (x2)
MSCR 582-583	32	— ⁽¹⁾	32 (x2)
MSCR 584-585	32	— ⁽¹⁾	32 (x2)
MSCR 586-587	32	— ⁽¹⁾	32 (x2)
MSCR 608	32	— ⁽¹⁾	32
MSCR 609	32	— ⁽¹⁾	32
MSCR 610	32	— ⁽¹⁾	32
MSCR 611	32	— ⁽¹⁾	32

Table 90. Protected SIUL2 registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MSCR 612	32	— ⁽¹⁾	32
MSCR 614	32	— ⁽¹⁾	32
MSCR 615	32	— ⁽¹⁾	32
MSCR 619	32	— ⁽¹⁾	32
MSCR 624	32	— ⁽¹⁾	32
MSCR 625	32	— ⁽¹⁾	32
MSCR 626	32	— ⁽¹⁾	32
MSCR 627	32	— ⁽¹⁾	32
MSCR 628	32	— ⁽¹⁾	32
MSCR 630	32	— ⁽¹⁾	32
MSCR 631	32	— ⁽¹⁾	32
MSCR 638	32	— ⁽¹⁾	32
MSCR 639	32	— ⁽¹⁾	32
MSCR 640	32	— ⁽¹⁾	32
MSCR 641	32	— ⁽¹⁾	32
MSCR 642	32	— ⁽¹⁾	32
MSCR 643	32	— ⁽¹⁾	32
MSCR 644	32	— ⁽¹⁾	32
MSCR 645	32	— ⁽¹⁾	32
MSCR 656-687	32	— ⁽¹⁾	32 (×32)
MSCR 688-719	32	— ⁽¹⁾	32 (×32)
MSCR 752-754	32	— ⁽¹⁾	32 (×3)
MSCR 758	32	— ⁽¹⁾	32
MSCR 759	32	— ⁽¹⁾	32
MSCR 760	32	— ⁽¹⁾	32
SIUL_GPDO0 - 511	8	0x1300	8 (×512)
SIUL_PGPDO0 - 31	16	0x1700	16 (×32)
SIUL_MPGPDO0 -31	32	0x1780	32 (×32)

1. See the module memory map in the SIUL2 chapter for address offset.

7.10.5.2 Core Clock Monitor Unit (CMU_CORE) protected registers

[Table 91](#) lists the registers that can be protected for the core CMU (CMU_CORE)

Table 91. Protected CMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_CSR	32	0x0	8 (byte 0)

7.10.5.3 I/O Processor Clock Monitor Unit (CMUIOP) protected registers

[Table 92](#) lists the registers that can be protected for the I/O Processor CMU (CMU_IOP)

Table 92. Protected CMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_CSR	32	0x0	8 (byte 0)

7.10.5.4 PMC Digital Interface (PMC_DIG) protected registers

[Table 93](#) lists the PMC_DIG registers can be protected.

Table 93. Protected PMC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
REE_VD3	32	0x0034	32
RES_VD3	32	0x0038	32
FEE_VD3	32	0x003C	32
REE_VD4	32	0x0044	32
RES_VD4	32	0x0048	32
FEE_VD4	32	0x004C	32
REE_VD7	32	0x0074	32
RES_VD7	32	0x0078	32
FEE_VD7	32	0x007C	32
REE_VD9	32	0x0094	32
FEE_VD9	32	0x009C	32
FEE_VD10	32	0x00AC	32
REE_VD12	32	0x00C4	32
RES_VD12	32	0x00C8	32
FEE_VD12	32	0x00CC	32
REE_VD13	32	0x00D4	32
RES_VD13	32	0x00D8	32
FEE_VD13	32	0x00DC	32

Table 93. Protected PMC_DIG registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
REE_VD14	32	0x00E4	32
RES_VD14	32	0x00E8	32
FEE_VD14	32	0x00EC	32
REE_VD15	32	0x00F4	32
RES_VD15	32	0x00F8	32
FEE_VD15	32	0x00FC	32
REE_TD	32	0x0304	32
RES_TD	32	0x0308	32
CTL_TD	32	0x030C	32
FEE_TD	32	0x0318	32
RES_VD9	32	0x098	32
REE_VD10	32	0x0A4	32
RES_VD10	32	0x0A8	32
VD_UTST	8	0x340	8
ADC_CH	8	0x344	8

7.10.5.5 PLL Digital Interface (PLL_DIG) protected registers

Table 94 lists the PLL_DIG registers that can be protected.

Table 94. Protected PLL_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PLL0_CR — PLL0 Control Register	32	0x0000	32
PLL0_DIVR — PLL0 Divider Register	32	0x0008	32
PLL1_CR — PLL1 Control Register	32	0x0020	32
PLL1_DIVR — PLL1 Divider Register	32	0x0028	32
PLL1_FMR — PLL1 Frequency Modulation Register	32	0x002C	32
PLL1_FDR — PLL1 Fractional Divide Register	32	0x0030	32

7.10.5.6 Oscillator Digital Interface (OSC_DIG) protected registers

[Table 95](#) lists the OSC_DIG registers that can be protected.

Table 95. Protected OSC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
OSC_CTL	32	0x0000	32

7.10.5.7 Memory Error Management Unit (MEMU) protected registers

[Table 96](#) lists the MEMU registers that can be protected.

Table 96. Protected MEMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTRL	8	0x000	8
DEBUG	32	0x00C	32

7.10.5.8 Mode Entry (MC_ME) protected registers

[Table 97](#) lists the MC_ME registers that can be protected.

Table 97. Protected MC_ME registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_MCTL	32	0x0004	32
ME_ME	32	0x008	32
ME_IM	32	0x010	32
ME_SAFE_MC	32	0x028	32
ME_DRUN_MC	32	0x02C	32
ME_RUN0_MC	32	0x030	32
ME_RUN1_MC	32	0x034	32
ME_RUN2_MC	32	0x038	32
ME_RUN3_MC	32	0x03C	32
ME_RUN_PC0–7	32	0x080–0x09C	32
ME_PCTL 0–255	8	0x0C4–0x1BF	8
ME_CCTL0–7	16	0x1C4–1D2	16

7.10.5.9 IMA (Indirect Memory Access) protected registers

[Table 98](#) lists the IMA register that can be protected.

Table 98. Protected IMA registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
IMA_ENABLE	32	0x004	32

7.10.5.10 Clock Generation Module (MC_CGM) protected registers

[Table 99](#) lists the MC_CGM registers that can be protected.

Table 99. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CGM_SC_DC[0..3]	32	0x07E8–0x7F4	32
CGM_AC[0..15]_SC	32	0x800–0x9E0	8 (byte 3)
CGM_AC[0..15]_DC[0..3]	32	0x808..814–0x9E8..9F4	32

7.10.5.11 Reset Generation Module (MC_RGM) protected registers

[Table 100](#) lists the MC_RGM registers that can be protected.

Table 100. Protected MC_RGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
RGM_DERD	32	0x010	32
RGM_DEAR	32	0x020	32
RGM_DBRE	32	0x030	32
RGM_FERD	32	0x310	32
RGM_FEAR	32	0x320	32
RGM_FBRE	32	0x330	32
RGM_FESS	32	0x340	32
RGM_PRSTn	32	0x610+n*0x4 ⁽¹⁾	32

1. n = 0, 1, 2, 3, 4, 5, 6, 7

7.10.5.12 System Status and Control Module (SSCM) protected registers

[Table 101](#) lists the SSCM registers that can be protected.

Table 101. Protected SSCM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ERROR	16	0x0006	16

7.10.5.13 Indirect Memory Access (IMA) protected registers

[Table 102](#) lists the IMA registers that can be protected.

Table 102. Protected IMA registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
IMA_ENABLE	32	0x0004	32

7.10.5.14 JTAG Master (JTAGM) protected registers

[Table 103](#) lists the JTAGM registers that can be protected.

Table 103. Protected JTAGM registers

Register	Register Size (bits)	Offset from Module Base Address	Protected Size (bits)
JTAGM_MCR	32	0x00	32

7.11 Security modules

The modules described in this section provide a basic set of customer-configurable security features designed to protect code and data from unauthorized access and to provide a way implement an audit trail to identify accesses to specific blocks of flash memory.

Table 104. Reference links to related information

Topic	Related module	Reference
Security overview	—	Chapter 11: Security
Password protection of flash memory blocks	PASS	Chapter 37: Flash Memory Programming and Configuration
	PASS	Chapter 84: Password and Device Security Module (PASS)
Tamper Detection	TDM	Chapter 85: Tamper Detection Module (TDM)
Detailed information on advanced security features	—	See the <i>SPC57EM80xx Microcontroller Security Reference Manual</i>

7.11.1 Password and Device Security Module (PASS) configuration

The PASS module is used to implement password-based read and write protection for flash blocks on the SPC57EM80xx MCU. Up to 4 levels of password protection can be implemented for each flash block.

Note: The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

See the Password and Device Security Module (PASS) chapter for further details.

7.11.1.1 PASS_LOCKn_PGn register bit mapping

As described in [Chapter 37: Flash Memory Programming and Configuration](#), each password group defined in the DCF records has a set of 4 LOCKn registers association with it—PASS_LOCKn_PGn. The bits of these registers are associated with specific flash blocks. The mapping is shown in the following figures.

Note: The mapping of various LOCK register bits to flash blocks in the SPC57EM80xx microcontroller is very similar, but not identical. The PASS module has additional mapping in the LOCK3 register and defines read locking regions. See the PASS chapter for details.

Figure 28. PASS_LOCK0_PGn Register⁽¹⁾

Bit Number	Register Field	Flash Block Name	Read Lock Region
0	TSLOCK	UTest NVM Block Space 16 KB	Read Lock Region 0
1	ATSL	—	—
2	LOWLOCK[13:0]	—	—
3		—	—
4		64 KB HSM Code block3	Read Lock Region 3
5		64 KB HSM Code block2	
6		64 KB Code Flash block1	Read Lock Region 1
7		64 KB Code Flash block0	
8		32 KB Code Flash block1	
9		32 KB Code Flash block0	
10		16 KB HSM Code block5	Read Lock Region 3
11		16 KB Code Flash block4	Read Lock Region 1
12		16 KB Code Flash block3	
13		16 KB Code Flash block2	
14		16 KB Code Flash block1	
15	MIDLOCK[15:0]	16 KB BAF block0	—
16		—	—
17		—	—
18		—	—
19		—	—
20		—	—
21		—	—
22		—	—
23		—	—
24		—	—
25		—	—
26		—	—
27		—	—
28		—	—
29		—	—
30		16 KB HSM EEPROM block1	Read Lock Region 4
31		16 KB HSM EEPROM block0	

1. Refer to [Section 37.4: Secure read protection](#) for details of flash read lock protection.

Figure 29. PASS_LOCK1_PGn Register⁽¹⁾

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Bit Name	ATSL	Reserved															HIGHLOCK[15:0]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
Read Lock Region	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Read Lock Region 2																																

1. Refer to [Section 37.4: Secure read protection](#) for details of flash read lock protection.

Figure 30. PASS_LOCK2_PGn Register⁽¹⁾

Bit Number	Register Bit Name	Flash block Name	Read Lock Region
0	256LCK_L[31:0]	—	—
1		—	—
2		—	—
3		—	—
4		—	—
5		—	—
6		—	—
7		—	—
8		—	—
9		—	—
10		—	—
11		—	—
12		—	—
13		—	—
14		—	—
15		—	—
16		—	—
17		—	—
18	Read Lock Region 1	256KB Code Flash block13	Read Lock Region 1
19		256KB Code Flash block12	
20		256KB Code Flash block11	
21		256KB Code Flash block10	
22		256KB Code Flash block9	
23		256KB Code Flash block8	
24		256KB Code Flash block7	
25		256KB Code Flash block6	
26		256KB Code Flash block5	
27		256KB Code Flash block4	
28		256KB Code Flash block3	
29		256KB Code Flash block2	
30		256KB Code Flash block1	
31		256KB Code Flash block0	

1. Refer to [Section 37.4: Secure read protection](#) for details of flash read lock protection.

Figure 31. PASS_LOCK3_PGn Register⁽¹⁾

Flash block Name	Register Bit Name	Bit Number
Password Group Lock	PGL	0
Debug Interface Lock	DBL	1
Master Only	MO	2
—	Reserved	3
Master Access	MSTR	4
		5
		6
		7
—	Reserved	8
—		9
—		10
HSM EEPROM Data Flash Read Lock	RL4	11
HSM Code Flash Read Lock	RL3	12
EEPROM Data Flash Read Lock	RL2	13
Code Flash Read Lock	RL1	14
UTEST Read Lock	RL0	15
—	256KLOCK_U[15:0]	
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		
—		

1. Refer to [Section 37.4: Secure read protection](#) for details of flash read lock protection.

7.11.1.2 PASS Module DCF Records (includes life cycle state and censorship)

The records required to configure the initial values for the PASS module password group registers is shown in [Table 105](#). These values provide the following capabilities:

- Set the current life cycle state
- Set flash memory to censored/uncensored state (censorship also depends on lifecycle)
- Override the basic flash write protection for flash memory blocks
- Set levels of password-based write protection (up to four 256-bit passwords can be required) for individual blocks of flash memory
- Set levels of password-based read protection (up to four 256-bit passwords can be required) for up to five flash memory Read Locking Regions
- Set levels of password-based protection (up to four 256-bit passwords can be required) for the debug part.

Table 105. PASS DCF records

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0000_1000	00000000	00101000	Life Cycle (LCSTAT) ⁽¹⁾	8	1
000_0000_0000_1000	00000000	00101001–00101011	Reserved		
000_0000_0000_1000	00000000	00101100	Censorship ⁽²⁾	16	1
000_0000_0000_1000	00000000	00101101–00101111	Reserved		
000_0000_0000_1000	00000000	00110000	Production Disable	2	1
000_0000_0000_1000	00000000	00110001–00111111	Reserved		
000_0000_0000_1000	00000000	01000000	LOCK0_PG0 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01000001	LOCK1_PG0 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01000010	LOCK2_PG0 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01000011	LOCK3_PG0 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01000100	LOCK0_PG1 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01000101	LOCK1_PG1 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01000110	LOCK2_PG1 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01000111	LOCK3_PG1 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01001000	LOCK0_PG2 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01001001	LOCK1_PG2 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01001010	LOCK2_PG2 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01001011	LOCK3_PG2 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01001100	LOCK0_PG3 ⁽³⁾	32	1

Table 105. PASS DCF records (continued)

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0000_1000	00000000	01001101	LOCK1_PG3 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01001110	LOCK2_PG3 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01001111	LOCK3_PG3 ⁽³⁾	32	1
000_0000_0000_1000	00000000	01001100–11111111	Reserved		

1. See [Section 84.3.1.1, Life Cycle Status register \(LCSTAT\)](#) for values.
2. See [Section 84.5.4, Censoring/uncensoring the device](#) for values.
3. See [Section 7.11.1.1, PASS_LOCKn_PGn register bit mapping](#) for values.

7.11.2 Tamper Detection Module (TDM) configuration

The Tamper Detection Module provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

Additionally, the TDM provides a mechanism for configuring individual flash memory blocks as One Time Programmable (OTP), i.e., write once, blocks.

The following sections provide details on the TDM configuration registers and DCF clients.

Warning: The TDM configuration registers detailed in this section are DCF clients. In general, DCF clients can only be written once—their intended values must be carefully selected beforehand.

See the Tamper Detection Module (TDM) chapter for further details on the TDM, including the DCF clients and two additional memory-mapped registers.

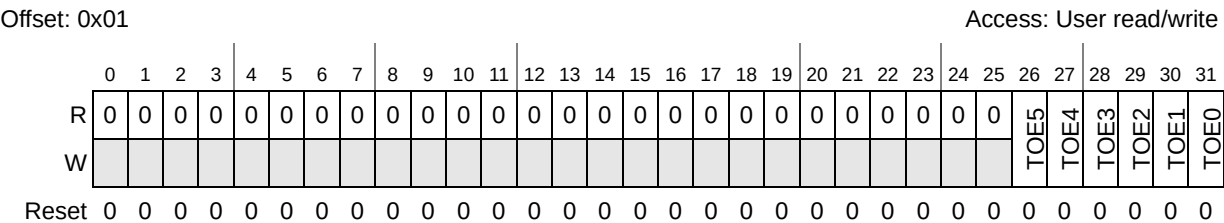
7.11.2.1 Diary Base Address (DBA) DCF client

DBA DCF client holds the base address of the diary, which is the region of flash that contains records corresponding to erase operations. This DCF client is more fully described in the TDM chapter.

7.11.2.2 Tamper Region Override (TO) DCF client

Tamper detection can be overridden by writing to enable bits (one per region) in the Tamper Region Override (TO) DCF client. This DCF client is more fully described in the TDM chapter. The mapping is shown below.

Figure 32. Tamper Region Override (TO) DCF client



The mapping of the Tamper Region Override Enable (TOEn) DCF client is as follows:

Table 106. TO[TOEn] field mapping

Field	TDR
TOE0	0
TOE1	1
TOE2	2
TOE3	3
TOE4	4
TOE5	5

This DCF client is more fully described in the TDM chapter

7.11.2.3 OTPENn DCF clients

Configuring individual flash blocks as OTP blocks is accomplished by setting the appropriate values in the TDM_OTPENn DCF clients. Each mapped bit is associated with a specific flash block. A “1” bit value indicates a block is OTP.

The mapping is shown in [Figure 28](#) - [Figure 36](#).



Figure 33. TDRx_LOCK0/OTPEN0 DCF client

Flash Block Name	Field	Bit Number
—	Reserved	0
—		1
—	LOWLOCK[13:0]	2
—		3
64 KB HSM Code block3		4
64 KB HSM Code block2		5
64 KB Code Flash block1		6
64 KB Code Flash block0		7
32 KB Code Flash block1		8
32 KB Code Flash block0		9
16 KB HSM Code block5		10
16 KB Code Flash block4		11
16 KB Code Flash block3		12
16 KB Code Flash block2		13
16 KB Code Flash block1		14
16 KB BAF block0		15
—	MIDLOCK[15:0]	16
—		17
—		18
—		19
—		20
—		21
—		22
—		23
—		24
—		25
—		26
—		27
—		28
—		29
16 KB HSM EEPROM block1		30
16 KB HSM EEPROM block0		31

Figure 34. TDRx_LOCK1/OTPEN1 DCF client

Bit Number	Bit Name	Flash block Name
0	Reserved	—
1		—
2		—
3		—
4		—
5		—
6		—
7		—
8		—
9		—
10		—
11		—
12		—
13		—
14		—
15		—
16	HIGHLOCK[15:0]	—
17		—
18		—
19		—
20		—
21		—
22		—
23		—
24		—
25		—
26		—
27		—
28		EEPROM Block3 (64KB EEPROM)
29		EEPROM Block2 (64KB EEPROM)
30		EEPROM Block1 (64KB EEPROM)
31		EEPROM Block0 (64KB EEPROM)



Figure 35. TDRx_LOCK2/OTPEN2 DCF client

Bit Number	Bit Name	Flash block Name
0	256KLOCK[31:0]	—
1		—
2		—
3		—
4		—
5		—
6		—
7		—
8		—
9		—
10		—
11		—
12		—
13		—
14		—
15		—
16		—
17		—
18		256KB Code Flash block13
19		256KB Code Flash block12
20		256KB Code Flash block11
21		256KB Code Flash block10
22		256KB Code Flash block9
23		256KB Code Flash block8
24		256KB Code Flash block7
25		256KB Code Flash block6
26		256KB Code Flash block5
27		256KB Code Flash block4
28		256KB Code Flash block3
29		256KB Code Flash block2
30		256KB Code Flash block1
31		256KB Code Flash block0

Figure 36. TDRx_LOCK3/OTPEN3 DCF client

Bit Number	Bit Name	Flash block Name
0	Reserved	—
1		—
2		—
3		—
4		—
5		—
6		—
7		—
8		—
9		—
10		—
11		—
12		—
13		—
14		—
15		—
16	256KLOCK[15:0]	—
17		—
18		—
19		—
20		—
21		—
22		—
23		—
24		—
25		—
26		—
27		—
28		—
29		—
30		—
31		—

7.11.2.4 TDRx_LOCKn DCF clients

Configuration of the 6 tamper regions is accomplished by setting field values in LOCK DCF clients (4 for each TDR).

Each set of 4 LOCKn DCF clients acts as a map of the MCU's flash memory blocks. A "1" in a LOCKn bit indicates that software must write a record before the corresponding block can be modified. The mapping of TDM_LOCKn DCF client bits to flash blocks is identical to the OTPENn mapping and is shown in [Figure 28](#) - [Figure 36](#). Any available flash block can be mapped to any tamper protection region.

7.11.2.5 DCF client details

[Table 107](#) contains the DCF client information.

Table 107. Tamper Detection Module (TDM) DCF clients

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00000000	Diary Base Address	32	1
000_0000_0001_0000	0000000	00000001	Tamper Region Override	6	1
000_0000_0001_0000	0000000	00000010 - 00000111	Reserved		
000_0000_0001_0000	0000000	00001000	OTPEN0 ⁽¹⁾	15	1
000_0000_0001_0000	0000000	00001001	OTPEN1 ⁽¹⁾	4	1
000_0000_0001_0000	0000000	00001010	OTPEN2 ⁽¹⁾	14	1
000_0000_0001_0000	0000000	00001011	OTPEN3 ⁽²⁾	0	1
000_0000_0001_0000	0000000	00001100-00010011	Reserved		
000_0000_0001_0000	0000000	00010100	TDR0_LOCK0	32	1
000_0000_0001_0000	0000000	00010101	TDR0_LOCK1	32	1
000_0000_0001_0000	0000000	00010110	TDR0_LOCK2	32	1
000_0000_0001_0000	0000000	00010111	TDR0_LOCK3	32	1
000_0000_0001_0000	0000000	00011000	TDR1_LOCK0	32	1
000_0000_0001_0000	0000000	00011001	TDR1_LOCK1	32	1
000_0000_0001_0000	0000000	00011010	TDR1_LOCK2	32	1
000_0000_0001_0000	0000000	00011011	TDR1_LOCK3	32	1
000_0000_0001_0000	0000000	00011100	TDR2_LOCK0	32	1
000_0000_0001_0000	0000000	00011101	TDR2_LOCK1	32	1
000_0000_0001_0000	0000000	00011110	TDR2_LOCK2	32	1
000_0000_0001_0000	0000000	00011111	TDR2_LOCK3	32	1
000_0000_0001_0000	0000000	00100000	TDR3_LOCK0	32	1
000_0000_0001_0000	0000000	00100001	TDR3_LOCK1	32	1

Table 107. Tamper Detection Module (TDM) DCF clients (continued)

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00100010	TDR3_LOCK2	32	1
000_0000_0001_0000	0000000	00100011	TDR3_LOCK3	32	1
000_0000_0001_0000	0000000	00100100	TDR4_LOCK0	32	1
000_0000_0001_0000	0000000	00100101	TDR4_LOCK1	32	1
000_0000_0001_0000	0000000	00100110	TDR4_LOCK2	32	1
000_0000_0001_0000	0000000	00100111	TDR4_LOCK3	32	1
000_0000_0001_0000	0000000	00101000	TDR5_LOCK0	32	1
000_0000_0001_0000	0000000	00101001	TDR5_LOCK1	32	1
000_0000_0001_0000	0000000	00101010	TDR5_LOCK2	32	1
000_0000_0001_0000	0000000	00101011	TDR5_LOCK3	32	1
000_0000_0001_0000	0000000	00101100 - 1111111	Reserved		

1. OTP DCF clients have the same structure and mapping as the write-protect bits of the flash and PASS module LOCK DCF clients. A value of '1' in an OTP bit indicates a block is programmable only once. Note that a block cannot be reconfigured as non-OTP after it has been configured as OTP.
2. There are no flash memory blocks mapped to the OTPEN3 DCF client.

7.12 Production device vs. emulation device

SPC57EM80xx microcontrollers have a comprehensive feature set to assist debug and calibration. Specialized variants, referred to as Emulation and Debug Devices (EDs), include additional calibration and debug features. An ED consists of the normal production die plus an additional die, called a Buddy die or Buddy Device (BD), bonded together in the same package. A Production Device (PD) includes a production die only and no buddy die.

Note: All features and modules included in the PD are also included in the ED.

The following table lists the key differences between the ED and PD of the SPC57EM80xx microcontroller.

Table 108. Debug and calibration features per SPC57EM80xx type

Debug or calibration feature	SPC57EM80xx PD	SPC57EM80xx ED
Internal Overlay Memory	16 KB	16 KB
Extended Overlay Memory	—	1 MB
Debug Interface	JTAG / LFAST	JTAG / LFAST
Trace Interface	—	Aurora (1.25 GHz, 4 Lane)
Trace Sources	Data trace on all bus accesses initiated from PD, Program trace on main CPUs, DMA and GTM	Data trace on all bus accesses initiated from PD, Program trace on main CPUs, DMA and GTM

Table 108. Debug and calibration features per SPC57EM80xx type (continued)

Debug or calibration feature	SPC57EM80xx PD	SPC57EM80xx ED
Trace Destination	Internal Overlay	Internal Overlay, Extended Overlay, Aurora Trace Port
Calibration Remap capabilities	32 Region	32 Region
System Level Performance Monitor Counters ⁽¹⁾	16 × 32 bits	16 × 32 bits
CPU Level Performance Monitor Counters	12 × 32 bits (4 × 32 bits per z4 CPU)	12 × 32 bits (4 × 32 bits per e200z4 CPU)

1. All system level PMCs can also be configured for use as debug timers.

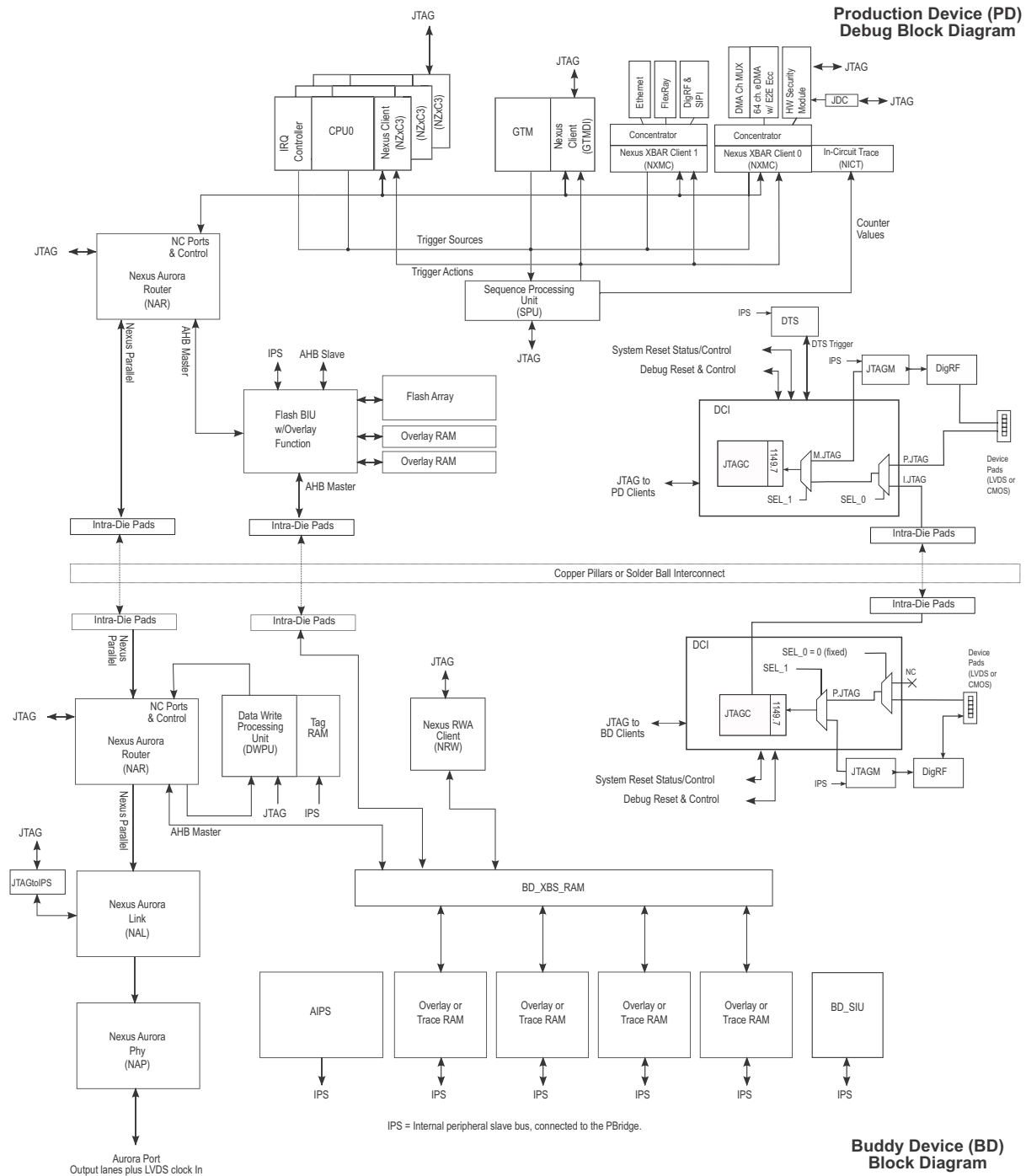
The following figure shows the top-level block diagram of the SPC57EM80xx ED, which consists of the production die packaged together with the 1 MB Buddy die (BD1M). The figure shows the general interconnection of functional debug modules including interconnections between the two devices. All functional blocks are shown for the BD, but for clarity only blocks relevant to the debug and calibration system are shown for the PD.

Acronyms used in the block diagram include:

- BD_SIU — BD System Integration logic
- BD_XBS_RAM — Buddy Device RAM Crossbar
- DCI — Debug and Calibration Interface
 - JTAGC — JTAG Controller
 - M.JTAG — Mastered JTAG
 - P.JTAG — Peripheral JTAG
 - I.JTAG — Inter-die JTAG
 - DCU — Debug Control Unit
- DTS — Development Trigger Semaphore
- JTAGM — JTAG Master
- SPU — Sequence Processing Unit
- NAR — Nexus Aurora Router
- NAL — Nexus Aurora Link
- Nexus RWA — Nexus Read/Write Access client

On ED devices, the device packaging routes the tool interface to the interface logic on the BD. On standard production devices packages the device packaging routes the tool interface to the interface logic on the PD.

Figure 37. Debug and calibration architecture for SPC57EM80xx ED



8 Reset and Boot

8.1 Introduction

Resetting the SPC57EM80xx involves an entire process starting with the application of power to actually fetching the first instruction of the user's application code. In general, the SPC57EM80xx is much more than a microcontroller with some selected group of peripheral elements and bus interfaces. It is an entire system containing multiple processing elements with their own associated local memory systems as well as memory elements that can be accessed by any of the processor cores.

To properly understand reset, some knowledge of the chip architecture is needed. There are five processor cores on the SPC57EM80xx. There are Core_0 and Core_1, both based on the e200_z420 architecture. These cores are used, in general, to execute user application code. Associated with Core_0 is Checker Core_0. This core executes the same instruction stream as Core_0. The results of each instruction execution are compared between the two cores and are flagged if a mismatch occurs. Peripheral Core_2, generally referred to as the IOP (or I/O Processor), is implemented using an e200_z425 core. Its intended purpose is to control the various peripheral elements on the SPC57EM80xx; however, there are no limitations to the types of programs that it can execute. Also included on the SPC57EM80xx is a Hardware Security Module (HSM) that has an embedded e200_z0 core. The HSM can execute code written by the factory and stored in a special Secure Flash that can only be accessed by the HSM. However, like the other processor cores, the HSM can access other memory elements on the SPC57EM80xx as well.

There are several memory elements that are also involved in the boot-up process. These are the UTEST flash memory area, the TEST flash memory area, and the Boot Assist Flash (BAF).

8.1.1 Boot Assist Flash

The Boot Assist Flash (BAF) contains code written by the manufacturer that facilitates the boot-up procedure of the SPC57EM80xx. Two important features of the BAF code are that it searches for a valid boot header and if none is found, the BAF code starts and runs the Serial Boot Loader. If there is no valid boot header loaded into the program memory when the SPC57EM80xx device is powered on, the Serial Boot Loader software can receive files from an external serial link controlled by the internal LINFlexD module. The received file is put into RAM and then program execution begins at the start address specified in the received file. The serial boot loader is only an option in the early stages of the device life cycle. In later life cycle stages (for example "field") the device simply waits for the watchdog to reset the device if no boot header is found.

8.1.2 TEST flash memory block

The TEST flash memory block contains Device Configuration Format (DCF) records used to hold trim values and other variables that are calculated during production test as well as general device configuration information. These trim values are for such things as adjusting low-voltage and high-voltage detect circuit trip points, temperature sensor adjustments, power-on reset voltage trip point, analog-to-digital adjustments and internal RC oscillator (IRCOSC) trim values. These values are determined during production testing and written into the TEST flash memory area. The ability to further program the TEST flash area is

disabled at the end of the factory test cycle. The DCF records are written by the device manufacturer and programmed into the TEST flash memory area during production testing.

8.1.3 UTEST flash memory block

The UTEST flash memory block also contains DCF records. Some UTEST DCF records are written by the factory and programmed during production testing. Others are written by the end user and programmed at the same time application code is programmed into the flash memory. UTEST DCF records are used to set up various control and configuration registers, including the Self-Test Control Unit (STCU2) and default memory configuration. UTEST DCF records, written by the device manufacturer, are programmed into the UTEST flash memory area during factory testing.

8.1.4 Boot header

A boot header is also needed in addition to the Device Configuration Records. A boot header is used to supply the start address for code execution for Core_0, Core_1 and Checker Core_0s. The boot header is written by the application programmer and loaded into specific locations in flash memory. The boot header contains the reset vectors for the various cores. The Boot Header also specifies the BOOT CPU that will eventually execute application code.

8.2 Modules used in reset sequence

There are several modules used to step through the sequence needed to properly reset the SPC57EM80xx and prepare it to fetch the first instruction of the user application code. These are the Power Management Controller (PMC), the Reset Generation Module (MC_RGM), the Mode Entry Module (MC_ME), the System Status and Configuration Module (SSCM) and the Self-Test Control Unit (STCU2).

8.2.1 Power Management Controller

The Power Management Controller (PMC) controls and monitors various voltage levels around the chip. Specifically, the PMC monitors its own supply voltage, the supply voltages to all the high- and low-voltage detect circuits, the trip points for all the high- and low-voltage detect circuits, the power supplies and reference voltages to the Analog-to-Digital Converter as well as the major power supplies to the SPC57EM80xx.

8.2.2 Reset Generation Module

The Reset Generation Module (MC_RGM) is a complex state machine that begins sequencing the SPC57EM80xx through the initial part of the reset process (see [Figure 38](#)). Reset sources are organized into two categories: destructive and functional. The MC_RGM is simply a state machine; it does not execute program code. In general, the reset generation module centralizes the different reset sources and manages the reset sequence.

8.2.2.1 Destructive resets

A destructive reset source is associated with an event related to a critical error or dysfunction, usually caused by a hardware malfunction. When a destructive reset event occurs, a software solution for recovery is not possible and the contents of memory are assumed to be unknown. Therefore, a full reset sequence is applied to the device starting

with PHASE0 in response to a destructive reset. This resets the full device ensuring a safe startup state for both digital and analog modules. See [Figure 38](#) for more information on the MC_RGM reset sequence.

Examples of destructive resets are:

- Power-on reset
- Low voltage detection
- Reset escalation (occurs when too many functional resets occur simultaneously)

8.2.2.2 Functional resets

A functional reset source is associated with an event related to a less critical error or dysfunction. These events are usually not associated with a hardware malfunction. When a functional reset event occurs, a partial reset sequence is applied to the device starting with PHASE1[FUNC]. See [Figure 38](#) for more information on the MC_RGM reset sequence. In this case, most digital modules are reset normally. The state of analog modules, specific digital modules (for example, debug and flash memory modules) and system memory content is preserved.

Examples of functional resets are:

- external reset
- machine check
- software reset from mode entry
- boundary scan instructions

8.2.3 Mode Entry module

The Mode Entry module (MC_ME) is responsible for delivering initial values to many registers in SPC57EM80xx. Among other things, the MC_ME supplies the reset vectors to all the SPC57EM80xx cores. During the power-up sequence, the System Status and Configuration Module (SSCM) searches for a boot header at defined locations in the flash memory, derives the necessary reset vectors from the boot header and then writes these vectors to their respective locations in the Mode Entry module. The SSCM also interprets the DCF records, which contain other information that is written to the MC_ME. As the boot-up sequence progresses, the SSCM instructs the MC_ME to transfer the reset vectors to the respective processor cores.

Even though the reset vectors are in the boot header located in flash memory, the Mode Entry module always provides them to the CPU cores.

8.2.4 System Status and Configuration Module

The System Status and Configuration Module (SSCM) is a state machine. During the reset sequence, the Reset Generation Module (MC_RGM) enables the SSCM, which reads the DCF records in both the TEST and UTEST flash memory areas. The DCF records contain device setup information that the SSCM uses to write data to various registers in the SPC57EM80xx. The DCF records are primarily concerned with setting up the memory, configuring the Self-Test Control Unit (STCU2), and providing initial device configuration values.

The SSCM continues with the reset or boot-up sequence by locating boot vectors for the various cores. Once the SSCM locates the boot header, the SSCM writes the start address information for the CPU cores to special locations in the Mode Entry Module (MC_ME). For

all cores enabled by the boot header, the MC_ME feeds a reset vector to each enabled core and the respective cores begin program execution at the specified address. The SSCM starts up the IOP and the HSM in different ways depending on whether there is valid code in the Boot Assist Flash (BAF). The boot-up sequence ends when the SSCM starts up the boot CPU, which can be either Core_0 or Core_1.

8.2.5 Self-Test Control Unit

The Self-Test Control Unit (STCU2) is a self-contained module that runs a Logic Built-In Self Test (LBIST) and a Memory Built-In Self Test (MBIST). The DCF record can be used to select which individual tests are run for LBIST and MBIST and disable tests for modules and memory elements that are not going to be used in user applications, thus shortening device boot-up time.

8.3 Reset sequence

The power-up reset sequence always begins with the application of power and follows different sequences depending on the condition of the SPC57EM80xx device and whether various modes are enabled from settings in the DCF records. For instance, the SPC57EM80xx device enters serial boot mode if a valid boot header file is not found. In this case, the device uses the serial boot loader to receive startup code and begin program execution. If an emulation and debug device is detected during the reset sequence, the device enters the calibration sequence. The calibration sequence is not a user mode and is only used for code development. Another reset outcome is that the Hardware Security Module, the IO Processor, and the boot CPU are all properly started and begin executing their respective program code.

8.3.1 Power-on and the Reset Generation Module

The initial power-up sequence starts with power being applied to the SPC57EM80xx. The Reset Generation Module (MC_RGM) then advances the device through a series of steps shown in [Figure 38](#). The remainder of the power-up sequence is detailed in [Figure 39](#) through to [Figure 43](#). The Reset Generation Module (MC_RGM) starts the System Status and Control Module (SSCM), which continues the boot-up process after the MC_RGM enters the IDLE state.

Figure 38. Reset Generation Module Reset Sequence

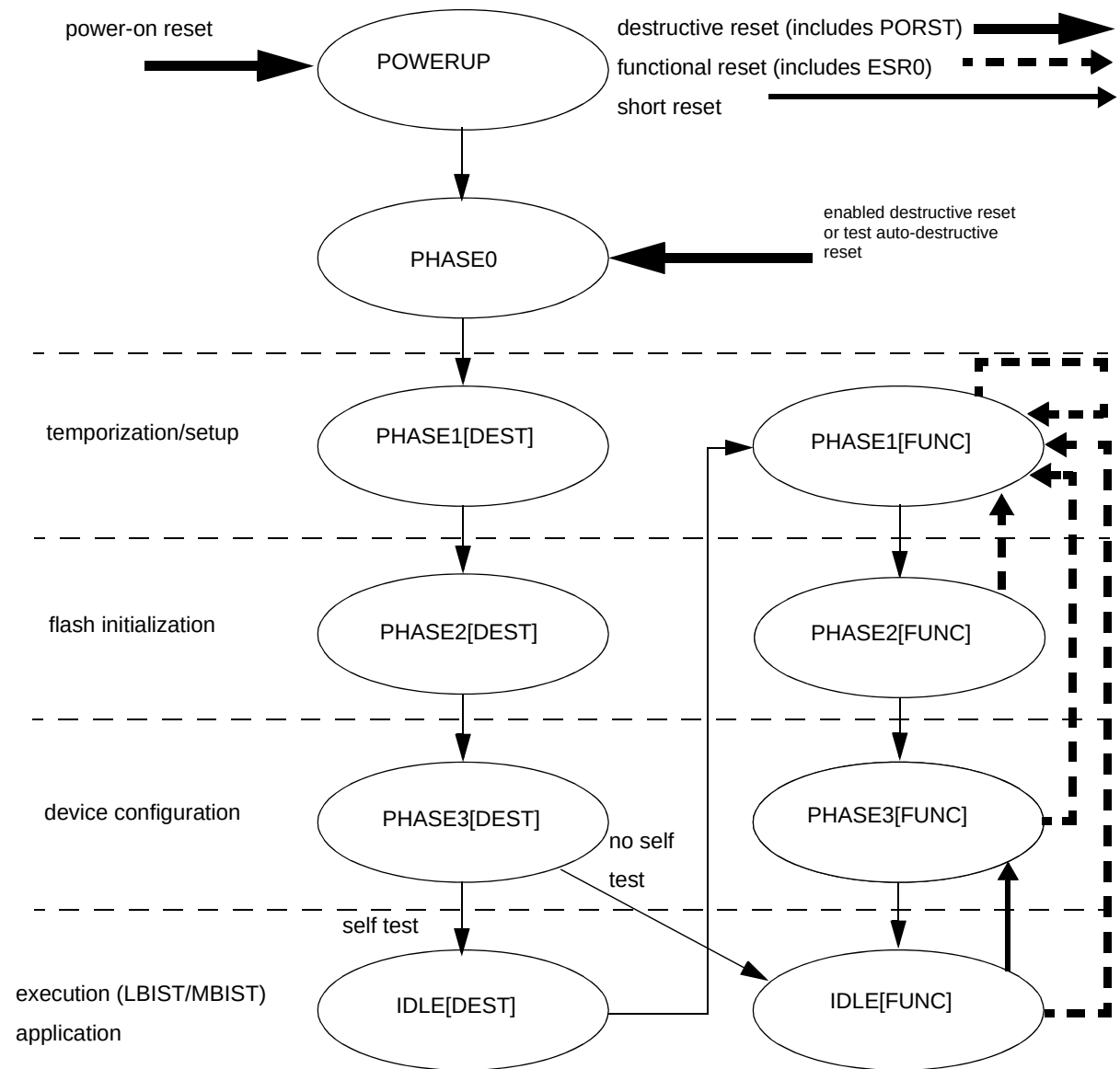


Table 109. Module status during reset phases

Module	PHASE								
	P0	P1 [DEST]	P2 [DEST]	P3 [DEST]	IDLE [DEST]	P1 [FUNC]	P2 [FUNC]	P3 [FUNC]	IDLE [FUNC]
IRCOSC	ON	ON	ON	ON	ON	ON	ON	ON	ON
PMC	ON	ON	ON	ON	ON	ON	ON	ON	ON
MC_RGM	ON ⁽¹⁾	ON	ON	ON	ON	ON	ON	ON	ON
System watchdog	RESET	RESET	ON	RESET	LBIST ⁽²⁾	RESET	ON	RESET	ON ⁽³⁾
Flash module	RESET	RESET	ON	ON	LBIST	RESET	ON	ON	ON
XOSC	RESET	RESET	RESET	ON ⁽³⁾	ON ⁽³⁾	ON ⁽³⁾	ON ⁽³⁾	ON ⁽³⁾	ON ⁽³⁾
DEVICE CONFIG	RESET	RESET	RESET	LOAD	HOLD	HOLD	HOLD	HOLD	HOLD
SSCM	RESET	RESET	RESET	ON	ON	RESET	RESET	ON	ON
FCCU	RESET	RESET	CONFIG	CONFIG	LBIST	RESET/ON ⁽⁴⁾	CONFIG/ON	CONFIG/ON	ON
STCU	RESET	RESET	CONFIG	CONFIG	ON	IDLE	IDLE	IDLE	IDLE ⁽⁵⁾
HSM	RESET	RESET	RESET	RESET	LBIST ⁽⁶⁾	RESET	RESET	RESET	ON
IOP	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON
PIT_1	RESET	RESET	RESET	RESET	LBIST	IDLE/ON	IDLE/ON	IDLE/ON	IDLE/ON
Other modules including cores	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON ⁽⁷⁾
								Short functional reset	
								functional reset	
						destructive reset			

1. RESET only during POR
2. STCU time-out monitoring LBIST/MBIST execution
3. Depending on flash configuration
4. FCCU remains ON during functional reset; it is instead reset in PHASE1[FUNC] if coming from IDLE[DEST]
5. Can be triggered by SW. Please refer to [Chapter 82: Self-Test Control Unit \(STCU2\)](#).
6. LBIST is optionally triggered depending on safety execution directives
7. LBIST is optionally triggered depending on safety execution directives after enabling from IOP. Please refer to [Chapter 58: Boot Assist Flash \(BAF\)](#).

8.3.2 Power-up phase: power stabilization

When a power-on reset event occurs (initial application of power), the reset generation module (MC_RGM) causes the device to enter the power-up phase of the reset sequence (See [Figure 38](#)). The only method to enter the power-up phase is from a power-on reset event.

The device remains in the power-up phase until the power management controller module (PMC) determines that all power supplies are in their respective operational ranges.

The supplies for initial configuration are:

- VDD_HV_CORE: high voltage supply for core module, including power management unit, internal RC oscillator (IRCOSC), external oscillator (XOSC) driver
- VDD_LV_CORE: low voltage supply for digital module
- VDD_HV_FLA: high voltage supply for flash module
- VDD_LV_FLA: low voltage supply for flash module
- VDD_HV_IO_IM: high voltage supply associated with system pins (PORST, ESR0, TEST Mode)

Peripheral supplies, such as VDD_HV_IO_JTAG, VDD_HV_IO_FLEX or VDD_HV_ADC, do not gate the power-up process. The low- and high-voltage detect circuits for these and other power supplies are managed by the PMC, which provides enable and status bits for these voltage detection circuits.

The PMC sends a signal to the Reset Generation Module when power stabilization has been achieved. The Reset Generation Module then advances to the PHASE0 state (see [Figure 38](#)). During the power-up phase, the PMC does the following:

- Drives to a logic 0 all LVDxx (low-voltage detect) and HVDxx (high-voltage detect) signals monitored by the Power Management Control (PMC) module.
- Monitors its own power supply voltage (VDD_HV_PMC) to determine that the applied voltage is within a specified range.
- Monitors the core power supply voltage (VDD_LV_CORE) to determine that the applied voltage is within a specified range.
- Monitors the low- and high-voltage detect circuit power supply voltage to determine that the applied voltage is within a specified range.
- Holds the outputs of the internal low- and high-voltage detect circuits at a ground state until the PMC determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Enables all low- and high-voltage detect circuits once all power supplies are at their functional levels.
- Begins to monitor the various power supplies using the high- and low-voltage detect circuits.
- Provides status information to the Reset Generation Module (MC_RGM) indicating whether or not power is properly applied.

To exit power-up and enter PHASE0:

- All enabled “destructive” resets must be processed.
- All power supplies must be at their functional voltages.
- The PMC must signal the MC_RGM that all power supplies are at the required levels.

8.3.3 **PHASE0 Phase: analog supply initial configuration**

This phase is entered:

- On exit from power-up phase.
- On PORST pin falling edge detection except for the initial power-up sequence.
- Immediately from PHASE2[DEST], PHASE3[DEST], IDLE[DEST], PHASE1[FUNC], PHASE2[FUNC], PHASE3[FUNC] or IDLE[FUNC], on the occurrence of a PORST falling edge event other than power-on reset.

During PHASE0:

- All digital modules are reset, including safety, security, and test modules.
- All trimming bits for analog modules are reset.
- Startup of the internal analog modules (PMC, IRCOSC, and I/Os) begins:
 - PMC determines that proper voltages are applied.
 - I/O pins are ensured correctly configured: outputs are driven to known levels, inputs ignored.

To exit PHASE0 and enter PHASE1[DEST]:

- All enabled destructive resets must be processed.
- Core Low Voltage Detect must be released.
- All processes initiated in **PHASE0** must be completed. This generally includes startup of the internal analog modules: PMC, IRCOSC, and I/Os.

This phase is completed in about 20 μ s.

Note: The core voltage must rise above LVD112 upper threshold to exit PHASE0. This ensures significant hysteresis on low voltage supply avoiding spurious low voltage detection during the power-up process.

8.3.4 **PHASE1[DEST] Phase: temporization and monitoring setup**

This phase is entered on exit from **PHASE0**.

During PHASE1[DEST]:

- Initial configuration of the watchdog to monitor the reset sequence
- Temporization for reset signal propagation in case of reset

This phase is completed in less than 1 μ s and corresponds to eight IRCOSC clock cycles.

During PHASE1[DEST], test modules are available. However, their actual test functionality is limited by the security module (no security waiver granted at this phase).

To exit PHASE1[DEST] and enter PHASE2[DEST], all PHASE1[DEST] tasks must be completed.

8.3.5 **PHASE2[DEST] Phase: flash initial configuration**

This phase is entered on exit from **PHASE1[DEST]**.

During PHASE2[DEST]:

- Reset is released to the flash memory module by the Reset Generation Module.
- The flash memory module starts its initial configuration process.
- The flash memory initialization is performed by a state machine internal to the flash module.
- The system watchdog starts monitoring flash memory configuration execution so that a reset occurs in the event of configuration failure.

Note: System Watchdog is configured to generate destructive reset in case of flash memory failure during boot process. If, on boot completion, the watchdog event is converted to generate an FCCU fault only, application software should re-configure the MC_RGM watchdog register bit to prevent destructive reset from being triggered.

To exit **PHASE2[DEST]** and enter **PHASE3[DEST]**, all processes started in **PHASE2[DEST]** must be completed.

This phase is completed in less than 200 μ s.

8.3.6 **PHASE3[DEST] Phase: device configuration**

This phase is entered on exit from **PHASE2[DEST]**.

During PHASE3[DEST]:

The System Status and Configuration Module (SSCM) is started. Configuration and initialization information in the Device Configuration Format (DCF) record is retrieved from the TEST and UTEST flash memory areas by the System Status and Configuration Module (SSCM) and used to configure the Self-Test Control Unit (STCU2) and the initial memory map of the device. It also transfers reset vectors for the various cores to the Mode Entry Module (MC_ME). This information enables tests that the STCU runs and also provides the memory map setup information so that the STCU can test memory. Trim values for the analog modules, such as the Analog-to-Digital Converter, low- and high-voltage detect circuits, and the temperature sensor, are written to their respective registers by the SSCM. The DCF record programmed into the TEST flash memory area is generally written by the manufacturer. The portion of the DCF record that is programmed into the UTEST flash memory is generally written by the customer although there is a default DCF record programmed into both the TEST and the UTEST flash memory areas during device production testing. In general, you write your own DCF record for the UTEST area to replace the default DCF.

The DCF record also provides configuration information for the following:

- Trimming of the analog modules (for example, voltage regulator, voltage detectors, I/Os, and oscillator)
 - Trim values are in DCF records in the TEST flash memory area.
 - Trim values are determined during production test and programmed into the TEST flash memory area.
- Device life cycle information:
 - Censorship configuration
 - Debug security configuration
 - Functional security configuration
- Application configuration bits:
 - Watchdog configuration
 - Safety execution directives (self test, LBIST/MBIST)
 - Boot Assist Flash options
 - Oscillator startup during reset sequence configuration

To exit **PHASE3[DEST]**: All processes that need to be completed in **PHASE3[DEST]** must be finished.

This phase is dependant on actual trim values, especially those used for safety execution directives. This device configuration is maintained until the next internal power-on reset. Each time a destructive reset event occurs, the SSCM takes values from the DCF record and write these values to the applicable registers.

If the configuration information from the DCF record requests that self test execution be performed, the device moves to **IDLE[DEST]** and the STCU2 begins self test execution. At the end of the self test execution, the STCU2 asserts a functional interrupt causing the Reset Generation Module to move to **PHASE1[FUNC]**. If self test execution is not requested, the device moves directly to **IDLE[FUNC]**. The reset generation module determines whether self test execution is enabled or not by examining an entry in the DCF record stored in the UTEST flash memory.

*Note: When self test execution is not requested, it is possible to maintain the device within **PHASE3[DEST]** by forcing either the PORST or ESR0 pin low. This can be used for debugging purposes by allowing external test/development equipment to connect through JTAG to the internal debugger.*

8.3.7 **IDLE[DEST] Phase: self test execution**

This phase is entered upon exit from **PHASE3[DEST]** if the execution of self test is enabled.

During the **IDLE[DEST]** phase:

- Dedicated watchdog is configured to monitor self test execution completion within expected time. The length of this phase is variable depending on the self-test requirements.
- The Self-Test Control Unit (STCU2) executes all tests specified by the DCF information retrieved in **PHASE3[DEST]**.
- The self test engine updates the self test completion flag and triggers an associated functional reset on completion of self test.

The length of this phase is variable depending on the self test requirements.

Note: When the SSCM is decoding the Device Configuration field, it sets and clears various flags in STCU registers and memory that can later be examined by the SSCM to determine which self test directives must be run.

Exiting IDLE[DEST] and moving to PHASE1[FUNC]

- The IDLE[DEST] phase is automatically exited at completion of self test when the self test code generates a functional reset.
- The functional reset causes the reset generation module to move to the **PHASE1[FUNC]** phase.

Note: In case of unrecoverable fault detection during STCU execution, unrecoverable fault reset are triggered and device re-enters **PHASE1[DEST]**.

8.3.8 **PHASE1[FUNC] Phase: temporization and monitoring setup**

This phase is entered either:

- On exit from **IDLE[DEST]**
- Immediately from **PHASE2[FUNC]**, **PHASE3[FUNC]**, or on **IDLE[FUNC]** when a non-masked external or “functional” reset event occurs, providing the source of the reset event has not been configured to trigger a ‘short’ reset sequence

During PHASE1[FUNC]:

- Initial configuration of the watchdog to monitor reset process
- Temporization for reset signal propagation in case of reset

To exit **PHASE1[FUNC]** and enter **PHASE2 [FUNC]**:

- All enabled, non-shortened “functional” resets must be processed.
- All processes started in **PHASE1[FUNC]** must be completed.
 - Initial configuration of the watchdog to monitor reset process
 - Temporization for reset signal propagation in case of reset

This phase is completed in ~1 µs or less. During this phase, test modules are available.

8.3.9 **PHASE2[FUNC] Phase: flash initial configuration**

This phase is entered on exit from **PHASE1[FUNC]**.

During the PHASE2[FUNC] phase:

- The reset signal is released to the flash memory.
- Flash initialization and configuration begins using a state machine that is internal to the flash memory module.
- A watchdog monitors flash initialization to ensure a reset in case of flash memory configuration failure.

The reset state machine exits **PHASE2[FUNC]** and enters **PHASE3[FUNC]** on verification that all processes started in **PHASE2[FUNC]** are completed.

This phase is completed in ~200 µs.

8.3.10 **PHASE3[FUNC] Phase: device configuration monitoring**

PHASE3[FUNC] is entered either:

- On exit from **PHASE2[FUNC]**
- Immediately from **IDLE[FUNC]** when an enabled, shortened functional reset event occurs

During **PHASE3[FUNC]**, configuration information contained within the UTEST flash memory sector is checked against information extracted during **PHASE3[DEST]** and written to the Self-Test Control Unit (STCU2). In case of mismatches, a dedicated boot fault is triggered within the FCCU which, if enabled, can cause a system reset. The purpose of **PHASE3[FUNC]** is to verify that the DCF information transferred to the STCU, MC_ME and other modules, in **PHASE3[DEST]**, was performed correctly and has not been corrupted.

The Reset Generation Module exits **PHASE3[FUNC]** and enters the **IDLE[FUNC]** on verification of the following:

- All processes started in **PHASE3[FUNC]** are completed.
- PORST and ESR0 pin are not forced low externally.
- A minimum number of cycles have elapsed since the last enabled, shortened “functional” reset event.

After all **PHASE3[FUNC]** internal actions have been completed, it is possible to maintain the device within **PHASE3[FUNC]** by forcing either the PORST or ESR0 pin low. This can be used for debugging purposes in order to connect through the JTAG port to the internal debugger. In the case of a secured device, it is possible to submit the debug password to access device and/or HSM debug through the JTAG module. The HSM responds by performing certification of the provided password and eventually may enable debug access to the different cores.

8.3.11 **IDLE[FUNC] Phase**

This is the Reset Generation Module's final phase of the reset sequence. It is entered either on exit from **PHASE3[DEST]**, if the execution of self test was not enabled, or at the completion of **PHASE3[FUNC]**. When the **IDLE[FUNC]** phase is reached, the MC_RGM releases control of the system to the System Status and Control Module. At this point, the SSCM which is stated in **PHASE3[FUNC]**, runs and waits for a signal from the MC_RGM indicating that the MC_RGM is now in the IDLE state. The MC_RGM then waits for new reset events that can trigger a reset sequence. The SSCM continues with the system boot-up sequence.

8.4 **MC_RGM passes boot sequence control to the System Status and Control Module**

The SSCM continues the boot-up sequence after the Self-Test Control Unit (STCU2) finishes executing the MBIST and LBIST functions that were initiated by the MC_RGM during the **IDLE[DEST]** state.

A flow chart of the boot-up sequence is shown in [Figure 39](#), [Figure 40](#), [Figure 41](#), [Figure 42](#), and [Figure 43](#).

8.4.1 Reset sequence flow based on initial device condition

The SPC57EM80xx is highly configurable. The flow through the boot-up sequence takes different paths depending on which modules are enabled and which header files are present. For instance, a device shipped from the factory has pre-installed programs in the Boot Assist Flash and the Boot Assist ROM. Also DCF records are programmed into the TEST flash memory area and the UTEST flash memory area.

- Boot Assist Flash
 - Contains code written by factory.
 - Cannot be modified by user.
- Boot Assist ROM
 - Contains code written by factory.
 - Cannot be modified by user.
- TEST DCF record cannot be modified by user
- UTEST DCF record
 - UTEST flash memory contains a default DCF record.
 - User-supplied UTEST DCF file can replace default UTEST DCF record.

After the SPC57EM80xx flash memory has been programmed, it contains the CPU boot header file, and, if required, an HSM boot header file, together with the user application. This boot header is used by the boot-up sequence flow which is managed by the SSCM module.

8.4.2 Possible boot-up sequences

8.4.2.1 Enter JTAG test mode

- Used for JTAG testing (non-user mode)
- Interface to JTAG port Development Equipment for Code Development

8.4.2.2 Enable IOP and enable the Boot CPU (CPU_0 or CPU_1)

- Normal operation for I/O Processor (IOP)
- Normal operation for the selected boot CPU

8.4.2.3 Enable IOP, HSM and enable the Boot CPU (CPU_0 or CPU_1)

- Normal operation for I/O Processor (IOP)
- Normal operation for selected boot CPU
- Normal Hardware Security Module (HSM) operation

8.4.2.4 Enable IOP and/or HSM and enter serial boot mode if no application code is present

- Mode used to download program code and place in RAM
- Mode used to start downloaded application program

The following sections detail the exact flow through the reset sequence to achieve the various outcomes.

8.4.3 Sequence from MC_RGM IDLE to IOP, HSM, and boot CPU running code

This path is taken when operation of both the IOP and HSM is desired. In this scenario, a boot CPU is selected and starts executing application code. Also, this is the scenario that is used to start all of the processor cores: CPU_0, CPU_1, Checker Core_0s, the HSM, and the IOP.

8.4.3.1 Initial conditions

- Boot Assist Flash bypass mode is NOT enabled (execute BAF code).
- HSM is enabled.
- Application code for the IOP is programmed into Boot Assist Flash memory.
- Application code for the HSM is programmed into flash memory.
- Application code for the boot CPU is programmed into flash memory.
- A valid boot header for CPU_0, CPU_1, and the Checker Core_0s has been programmed into flash memory.
 - Boot Header_ID
 - Boot CPU selection
 - Boot CPU start address (address of first instruction in application code)
 - CPU_0 reset vector
 - CPU_1 reset vector
 - Checker Core_0s reset vector
- A valid boot header for HSM has been written into flash memory
 - HSM_Boot_Header_ID
 - Start address of HSM application code
- The Reset Generation Module (MC_RGM) has passed control to the System Status and Control Module during the IDLE[FUNC] state.

8.4.3.2 Reset Generation Module has entered idle state

The MC_RGM enters the IDLE[FUNC] state. The MC_RGM completes all the tasks in defined the IDLE[FUNC] state. Then, the MC_RGM signals the SSCM that the MC_RGM has become idle and that the SSCM is to take over control of the boot-up sequence.

8.4.3.3 LBIST and MBIST start execution (state 1 and 2)

The MC_RGM enters the IDLE[FUNC] state and completes all tasks. Then, the MC_RGM signals the SSCM that the MC_RGM has become idle and that the SSCM is to take over control of the boot-up sequence. At this point, LBIST and MBIST have completed their test cycles, the data in the DCF record has been written to the appropriate registers, and all the trim values for the analog portions of the chip have been installed in their proper locations.

8.4.3.4 Is a JTAG request pending? (state A)

The SSCM is now in charge of the boot-up sequence. It begins by determining if a JTAG request is pending. The JTAG test circuitry is for production test and use by software development systems. The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

8.4.3.5 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether or not the Boot Assist Flash bypass mode is enabled. If this mode is enabled, any eventual code in the BAF is not executed. If the Boot Assist Flash bypass mode is not enabled, the code in the BAF is executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the BAF bypass mode. Therefore, this mode is not available for customer use. With the BAF bypass mode disabled, code in the Boot Assist Flash is always be executed during a boot-up sequence.

The Boot Assist Flash code is written by the device manufacturer. Among other things, the BAF code runs the Serial Boot Loader. This involves setting up pins on the device to act as a UART, receiving a file using a defined format and transferring the program portion of the file to RAM. Once the program is installed, the SSCM causes the IOP (I/O Processor) to begin execution at the start address specified in the header portion of the program file.

8.4.3.6 Is the Hardware Security Module enabled? (state C)

The SSCM examines a status bit to determine if the Hardware Security Module (HSM) is enabled. A DCF record in the TEST area is responsible for enabling/disabling the HSM. In the present example, it is assumed that the HSM is enabled.

8.4.3.7 Locate HSM Boot Header (state 44)

The SSCM is responsible for locating the HSM boot header. The SSCM reads several memory locations in flash memory to determine whether a valid HSM boot header is present. Specifically, the SSCM reads locations 0x60_C000, 0x61_0000, and 0x62_0000. A valid HSM boot header consists of three words: an HSM_Boot_Header_ID, a start address for the HSM application code, and a third word that is reserved for future use. By definition a valid HSM_Boot_Header_ID is 0xFFFF_0000_FFFF_0000.

If a valid HSM_Boot_Header_ID is found, the SSCM transfers the start address to the Mode Entry Module (MC_ME). When reset is released to the HSM, the HSM start address is transferred from the MC_ME to the HSM and HSM code execution begins at that point.

8.4.3.8 Valid Hardware Security Module header file? (state D)

In the present scenario, a valid HSM boot header is present as determined in state 44.

8.4.3.9 Provide reset vectors to IOP and HSM (state 25—go to state 22 and 26)

Under the control of the SSCM, the HSM start address obtained from the HSM boot header is transferred to the MC_ME.

During PHASE3[DEST] of the Reset Generation Module's sequence, the start address of the BAF code loaded into the Boot Assist Flash is transferred to the MC_ME. The SSCM releases reset to the IOP and HSM and then turns on the clock to both modules. The BAF code start address is transferred from the MC_ME to the IOP. At this time, the HSM code start address is also transferred from the MC_ME to the HSM under the control of the SSCM.

8.4.3.10 HSM code and BAF code execution begins (state 26)

After the reset signal is released and the respective clock is turned on to the HSM and the IOP, code execution begins. The HSM begins executing its application code and the IOP begins execution of code in the Boot Assist Flash.

8.4.3.11 Verification of HSM host boot block (state 27)

HSM applications code can optionally be written to confirm that the correct start address for the HSM applications code was obtained from the HSM boot header file. This step is optional and requires that the HSM applications code and the HSM boot header file match.

If a new revision of the HSM code is created with a start address that is different from the prior revision, the HSM applications code must be modified to take into account that there is a different start address. The HSM Boot Header must also be appropriately modified.

8.4.3.12 Set HSM Boot Ready flag (state 28)

The HSM hardware sets the HSM Boot Ready flag when the HSM becomes fully initialized.

8.4.3.13 Execution of HSM application code begins (state 29)

The HSM has been properly enabled, a valid start address for code execution has been supplied and the HSM begins normal code execution.

8.4.3.14 IOP begins BAF code execution (state 22)

The BAF code begins by initializing registers and applying device settings. The BAF code is written by the device manufacturer and changes from device to device.

8.4.3.15 IOP applies device settings (state 23)

The IO Processor, executing code in the Boot Assist Flash, configures the SPC57EM80xx with various initialization parameters. The System Status and Configuration Module (SSCM) has already applied some configuration information from the DCF record.

8.4.3.16 Calibration Sequence (state 24)

This state is used to detect the presence of an emulation and debug device (ED) normally used in place of a regular production SPC57EM80xx (PO). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The emulation and debug device (ED) consists of a regular production SPC57EM80xx (PO) and a special Buddy Device (BD) in a single package.

The I/O Processor (IOP) executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is detected, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of SPC57EM80xx reset.

In this example, it is assumed that an emulation and debug device (ED) is not being used in place of a regular production SPC57EM80xx.

8.4.3.17 Is HSM enabled? (decision point E)

The IOP, executing BAF code, checks to see if the HSM is enabled. In this example, it is assumed that the HSM is enabled.

8.4.3.18 Wait for HSM to set HSM_Boot_Ready flag? (decision point F)

The IOP determines whether it must wait until the HSM becomes ready before the IOP begins a search for the boot header file. If the IOP is directed to wait for the HSM to set its

HSM_Boot_Ready flag, the IOP goes to state 30 until the “HSM_Boot_Ready” flag is set. Afterwards, the IOP begins a search for the Boot Header file in state 31.

If the IOP is not directed to wait for the HSM to set its HSM_Boot_Ready flag, the IOP immediately begins a search for the boot header file in state 31.

8.4.3.19 Wait until HSM_Boot_Ready flag sets (state 30)

The IOP waits until the HSM_Boot_Ready flag is set.

8.4.3.20 IOP searches for boot header (state 31)

The IOP continues code execution by searching for the boot header. This is the header file for the CPU that starts executing the your application code. Generally, the BOOT CPU is either Core_0 or Core_1.

The Boot CPU header is programmed into flash memory at the same time your application program is programmed into flash memory. The IOP searches for the boot header file which can be located at one of eight addresses:

Note: Addresses are searched in the following order.

- 0xFC_0000
- 0xFC_4000
- 0xFC_8000
- 0xFC_C000
- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid boot header begins with the value 0x005A.

The boot header structure consists of seven words:

- Boot Header_ID/Boot_CPU
- Start address
- Configuration bits
- Configuration bits
- CPU_0 reset vector
- CPU_1 reset vector
- Checker Core_0s reset vector

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

Only four bits in the 16-bit Boot_CPU field are used. Bit 0 selects the I/O Processor, bit 1 selects CPU_0, bit 2 selects CPU_1, and bit 3 selects the Checker Core_0s. Bits 4–15 are not implemented.

Note: Checker Core_0s can only be used if CPU_0 is also used. Both cores must have the same reset vector.

8.4.3.21 Valid boot header is found (state N)

In the present example, a valid boot header is assumed to be present in flash memory.

8.4.3.22 Decode boot CPU (state 32)

The IOP decodes the boot header and then writes the reset vectors for CPU_0, CPU_1, and the Checker Core_0s to the Mode Entry Module.

8.4.3.23 Lock BAF access (state 33)

Further access to the BAF is locked against further code execution, but it is readable.

8.4.3.24 Boot From Internal flash (state 34)

The MC_ME transfers the reset vectors to the respective CPUs and code execution for each of the cores begins at their respective reset vector address.

8.4.4 Sequence from MC_RGM IDLE to IOP and Boot CPU running code (HSM disabled)

This path is taken when operation of the IOP, but not the HSM, is desired. In this scenario, a Boot CPU is selected and starts executing application code. Also, this is the scenario that is used to start all of the processor cores except the HSM: CPU_0, CPU_1, Checker Core_0s, and the IOP.

8.4.4.1 Assumptions

- Boot Assist Flash bypass mode is not enabled (execute BAF code).
- HSM is not enabled.
- Application code for the IOP is programmed into Boot Assist Flash memory.
- Application code for the Boot CPU is programmed into flash memory.
- A valid boot header for CPU_0, CPU_1 and Checker Core_0s has been programmed into flash memory.
 - Boot Header_ID
 - Boot CPU selection
 - Boot CPU start address (address of first instruction in application code)
 - CPU_0 reset vector
 - CPU_1 reset vector
 - Checker Core_0s reset vector
- The MC_RGM has passed control to the SSCM during the IDLE[FUNC] state.

8.4.4.2 Reset Generation Module has entered idle state (state 1 and 2)

The Reset Generation Module (MC_RGM) enters the IDLE[FUNC] state and completes all tasks. Then, the MC_RGM signals the System Status and Control Module (SSCM) that the MC_RGM has become idle and that the SSCM is to take over control of the boot-up sequence. At this point, LBIST and MBIST have completed their test cycles, the data in the DCF record has been written to the appropriate registers and all the trim values for the analog portions of the chip have been installed in their proper locations.

8.4.4.3 Is a JTAG request pending? (state A)

The SSCM is now in charge of the boot-up sequence. It begins by determining whether a JTAG request is pending. The JTAG test circuitry is for production test and used by software development systems. The JTAG mode has no user functions and should never be used in

a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

8.4.4.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether the Boot Assist Flash bypass mode is enabled. If this mode is enabled, any eventual code in the BAF is not executed. If the Boot Assist Flash bypass mode is not enabled, the code in the BAF is executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. Therefore, this mode is not available for customer use. This means that the code in the Boot Assist Flash is always executed during a boot-up sequence.

The Boot Assist Flash code is written by the device manufacturer. Among other things, the BAF code runs the serial boot loader. This involves setting up pins on the device to act as a UART, receiving a file, using a defined format, and transferring the program portion of the file to RAM. Once the program is installed, the SSCM causes the IOP to begin execution at the start address specified in the header portion of the program file.

8.4.4.5 Is the Hardware Security Module enabled? (state C)

The SSCM examines a status bit to determine if the HSM is enabled. A DCF record in the TEST area is responsible for enabling/disabling the HSM. In the present example, it is assumed that the HSM is not enabled.

8.4.4.6 Provide reset vectors to IOP (state 21)

During PHASE3[DEST] of the Reset Generation Module's sequence, the start address for execution of the BAF code loaded into the Boot Assist Flash is transferred to the MC_ME.

The SSCM releases reset to the I/O Processor (IOP) and then turns on the clock to the IOP. The BAF code start address is transferred to the IOP at this time under the control of the SSCM. Execution of the Boot Assist Flash code begins.

8.4.4.7 BAF code execution begins (state 22)

After the reset signal is released to the IOP, BAF code execution begins.

8.4.4.8 The IOP executes code to apply device setting (state 23)

The IOP executes BAF code, written by the manufacturer, to apply specific device settings to the SPC57EM80xx.

The BAF code begins by initializing registers and applying device settings. The BAF code, written by the device manufacturer, changes from device to device.

8.4.4.9 Calibration sequence (state 24)

This state is used to detect the presence of an emulation and debug device (ED) normally used in place of a regular production SPC57EM80xx (PO). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The ED consists of a PO and a special Buddy Device (BD) in a single package.

The IOP executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is present, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of SPC57EM80xx reset.

In this example, it is assumed that an ED is not being used in place of a regular SPC57EM80xx production device.

8.4.4.10 Is HSM enabled? (decision point E)

The IOP determines whether the HSM is enabled by examining a status bit in the HSM. In the present example, it is assumed that the HSM is not enabled.

8.4.4.11 IOP searches for boot header (state 31)

The IOP continues code execution by searching for the boot header. In the present example, it is assumed that a valid Boot Header is present.

This is the header file that specifies which Boot CPU starts executing the user application code. Generally, the Boot CPU is either Core_0 or Core_1. The boot header file also contains the reset vector addresses for Core_0, Core_1, and the Checker Core_0s.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The IOP searches for the boot header, which can be located at one of eight addresses:

- 0xFC_0000
- 0xFC_4000
- 0xFC_8000
- 0xFC_C000
- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of seven words:

- Boot Header_ID/Boot_CPU
- Start address
- Configuration bits
- Configuration bits
- CPU_0 reset vector
- CPU_1 reset vector
- Checker Core_0s reset vector

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

Only four bits in the 16-bit Boot_CPU field are used. Bit 0 selects the I/O Processor, bit 1 selects CPU_0, bit 2 selects CPU_1, and bit 3 selects the Checker Core_0s. Bits 4–15 are not implemented.

8.4.4.12 Is a valid boot header present? (decision point N)

In the present example, a valid boot header is assumed to be present in flash memory.

8.4.4.13 Decode boot CPU (state 32)

The I/O Processor decodes the boot header and then writes the reset vectors for CPU_0, CPU_1, and the Checker Core_0s to the Mode Entry Module (MC_ME).

8.4.4.14 Lock BAF access (state 33)

Further access to the Boot Assist Flash is locked.

8.4.4.15 Boot from internal flash (state 34)

The MC_ME transfers the reset vectors to the respective CPUs and code execution for each of the cores begins at their respective reset vector.

8.4.5 Path from MC_RGM IDLE to serial boot mode

This path is taken when no application program or valid boot headers for the HSM or CPU_0, CPU_1, or Checker Core_0s is programmed into flash memory. This case is taken when a device has not been programmed with a user application. This is an unusual case in that it describes an improperly programmed device. It is shown here for illustration.

8.4.5.1 Initial conditions

- No valid boot header for CPU_0, CPU_1, and Checker Core_0s has been programmed into flash memory.
- BAF bypass mode is NOT enabled (execute BAF code).
- The HSM is enabled.
- Application code for the IOP is programmed into the BAF.
- Application code for the HSM is not programmed into secure flash memory.
- Application code for the Boot CPU is not programmed into flash memory.
- A valid boot header for the HSM has not been written into flash memory.
- The Life Cycle variable is set to FACTORY, indicating an unprogrammed device.
- The MC_RGM has passed control to the SSCM during the IDLE[FUNC] state.

8.4.5.2 Reset Generation Module has entered idle state (state 1 and 2)

The MC_RGM enters the IDLE[FUNC] state and completes all tasks. Then, the MC_RGM signals the SSCM that the MC_RGM has become idle and that the SSCM is to take over control of the boot-up sequence. At this point, LBIST and MBIST have completed their test cycles, the data in the DCF record has been written to the appropriate registers and all the trim values for the analog portions of the chip have been installed in their proper locations.

8.4.5.3 Is a JTAG request pending? (state A)

The SSCM is now in charge of the boot-up sequence. It begins by determining if a JTAG request is pending. The JTAG test circuitry is for production test and use by software development systems. The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

8.4.5.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether the BAF bypass mode is enabled. If this mode is enabled, any eventual code in the BAF is not executed. If the BAF bypass mode is not enabled, the code in the BAF is executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. Therefore, this mode is not available for use. This means that the code in the BAF is always executed during a boot-up sequence.

The Boot Assist Flash code is written by the device manufacturer. Among other things, the BAF code runs the serial boot loader. This involves setting up pins on the device to act as a UART, receiving a file, using a defined format, and transferring the program portion of the file to RAM. Once the program is installed, the SSCM causes the IOP to begin execution at the start address specified in the header portion of the program file.

8.4.5.5 Is the Hardware Security Module enabled? (state C)

The SSCM examines a status bit to determine if the HSM is enabled. A DCF record in the TEST area is responsible for enabling/disabling the HSM. In the present example, it is assumed that the HSM is enabled.

8.4.5.6 Locate HSM boot header (state 44)

The SSCM is responsible for locating the HSM boot header. The SSCM reads several memory locations in flash memory to determine whether a valid HSM boot header is present. Specifically, the SSCM reads locations 0x60_0000, 0x60_4000, and 0x61_4000. A valid HSM boot header consists of three words: an HSM_Boot_Header_ID, a Start Address for the HSM application code, and a third word that is reserved for future use. By definition, a valid HSM_Boot_Header_ID is 0xFFFF_0000_FFFF_0000.

8.4.5.7 Valid Hardware Security Module header file? (state D)

In the present scenario, no valid HSM boot header is present. In this case, the System Status and Control Module takes the same path as if the HSM was not enabled at decision point C.

8.4.5.8 Provide reset vectors to IOP (state 21)

During PHASE3[DEST] of the Reset Generation Module's sequence, the start address for execution of the BAF code in the Boot Assist Flash is transferred from the DCF record to the MC_ME by the SSCM.

The SSCM releases reset to the IOP and then turns on the clock to the IOP. The clock to the HSM is turned off in this state even though the HSM was originally enabled. Because no valid header file for the HSM is found, the IOP goes to state 21 from decision point D, which turns the HSM clock off. The start address is transferred to the IOP at this time under the control of the SSCM and the IOP begins execution of the BAF code.

8.4.5.9 IOP begins BAF code execution (state 22)

After the reset signal is released and the respective clocks are turned on to the IOP, the IOP begins execution of code in the Boot Assist Flash.

8.4.5.10 IOP continues BAF code execution (state 23)

The BAF code begins by initializing registers and applying device settings. The BAF code is written by the device manufacturer and changes from device to device.

8.4.5.11 Calibration Sequence (state 24)

This state is used to detect the presence of an emulation and debug device (ED) normally used in place of a regular production SPC57EM80xx (PO). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The ED consists of a PO and a special Buddy Device (BD) in a single package.

The IOP executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is present, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of SPC57EM80xx Reset.

In this example, it is assumed that an ED is not being used in place of a regular SPC57EM80xx production device.

8.4.5.12 IOP checks to determine if HSM is enabled (state E)

The IOP, executing BAF code, checks to see if the HSM is enabled. In the present case, the SSCM has already disabled the clock to the HSM. Therefore, the HSM is not enabled.

8.4.5.13 IOP searches for boot header (state 31)

The IOP continues code execution by searching for the boot header. In the present example, it is assumed that no boot header file is present.

This is the header file for the CPU that starts executing the user application code. Generally, the BOOT CPU is either Core_0 or Core_1.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The IOP searches for the boot header, which can be located at one of eight addresses:

- 0xFC_0000
- 0xFC_4000
- 0xFC_8000
- 0xFC_C000
- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of seven words:

- Boot Header_ID/Boot_CPU
- Start address
- Configuration bits
- Configuration bits
- CPU_0 reset vector
- CPU_1 reset vector
- Checker Core_0s reset vector

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

Only four bits in the 16-bit Boot_CPU field are used. Bit 0 selects the I/O Processor, bit 1 selects CPU_0, bit 2 selects CPU_1, and bit 3 selects the Checker Core_0s. Bits 4–15 are not implemented.

8.4.5.14 Valid boot header is not found (decision point N)

The IOP, executing code in the Boot Assist Flash, does not find a valid boot header file in flash memory. Because there is no valid boot header, serial boot mode must be entered to obtain an application program.

Note: This mode receives a program to be executed over a serial port implemented using the LINFlexD port. Once the program is received, using a defined format, the IOP jumps to the start address of the program and begins execution.

8.4.5.15 Initialize RAM (state 35)

The IOP, executing code in the Boot Assist Flash, initializes RAM in preparation for entering serial boot mode. Serial boot mode configures the LINFlexD port to receive code from an external source. The LINFlexD port comes in a number of hardware configurations. The SPC57EM80xx uses LINFlexD, which implements a basic UART. Code in the BAF operates the LINFlexD module and implements the protocol.

8.4.5.16 Enter serial boot mode (state 14)

[Figure 43 \(Boot-up sequence, part E\)](#) shows the boot loader flow.

8.4.5.17 Life Cycle check indicates FACTORY (decision point G)

There are two possible settings for the Life Cycle parameter—FIELD and FACTORY. A setting of FIELD indicates that application code has been programmed into the flash memory of the device. In this case, serial boot mode is exited. The IOP enters a “wait mode” and remains there until a watchdog reset occurs. A setting of FACTORY means that there is no application code programmed into the flash memory. For the present example a setting of FACTORY is assumed so there is no application code present.

8.4.5.18 Receive and install application code

In general, a serial link is implemented using the LINFlexD port and a particular protocol for data transmission has been defined and implemented in the Boot Assist Flash code. Once the LINFlexD port is configured, the serial boot loader waits to receive the code.

Note: There is a software watchdog timer that is started by the BAF code. If a program is not received before the watchdog timer timeout, a reset occurs.

- Configure LINFlexD module and external pins (state 37).
- Wait for START WORD (state 38).
- Receive START WORD (state 39).
- Receive START ADDRESS (state 40).
- Receive DOWNLOAD SIZE (state 41).
- Receive CODE (state 42).
- Branch to START ADDRESS (state 43).

Application code, using the IOP, begins at the received START ADDRESS.

8.4.6 Path from MC_RGM IDLE with IOP and HSM enabled to watchdog timer timeout

This path is taken when the operation of the IOP and the HSM is desired but there is no valid boot header file or HSM header file and the Life Cycle variable is set to FIELD.

The Life Cycle variable of FIELD indicates that the device was programmed with application code and the appropriate boot header files. This particular path is taken only if a catastrophic event has occurred or the device is improperly programmed.

8.4.6.1 Initial conditions

- Boot Assist Flash bypass mode is not enabled (execute BAF code).
- The HSM is enabled.
- A valid boot header for the HSM has not been programmed into flash memory.
- A valid boot header for CPU_0, CPU_1 and Checker Core_0s has not been programmed into flash memory.
 - Boot Header_ID
 - Boot CPU selection
 - Boot CPU start address (address of first instruction in application code)
 - CPU_0 reset vector
 - CPU_1 reset vector
 - Checker Core_0s reset vector
- The Life Cycle variable is set to FIELD, indicating a programmed device.
- The MC_RGM has passed control to the SSCM during the IDLE[FUNC] state.

8.4.6.2 Reset Generation Module has entered idle state (state 1 and 2)

The MC_RGM enters the IDLE[FUNC] state and completes all tasks. Then, the MC_RGM signals the SSCM that the MC_RGM has become idle and that the SSCM is to take over control of the boot-up sequence. At this point, LBIST and MBIST have completed their test cycles, the data in the DCF record has been written to the appropriate registers, and all the trim values for the analog portions of the chip have been installed in their proper locations.

8.4.6.3 Is a JTAG request pending? (state A)

The SSCM is now in charge of the boot-up sequence. It begins by determining if a JTAG request is pending. The JTAG test circuitry is for production test and use by software

development systems. The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

8.4.6.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether or not the boot assist flash bypass mode is enabled. If this mode is enabled, any eventual code in the Boot Assist Flash is not executed. If the Boot Assist Flash bypass mode is not enabled, the code in the BAF is executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. Therefore, this mode is not available for use. This means that the code in the BAF is always executed during a boot-up sequence.

The BAF code is written by the device manufacturer. Among other things, the BAF code runs the serial boot loader. This involves setting up pins on the device to act as a UART, receiving a file, using a defined format, and transferring the program portion of the file to RAM. Once the program is installed, the SSCM causes the IOP to begin execution at the start address specified in the header portion of the program file.

8.4.6.5 Is the Hardware Security Module enabled? (state C)

The SSCM examines a status bit to determine if the HSM is enabled. A DCF record in the TEST area is responsible for enabling/disabling the HSM. In the present example, it is assumed that the HSM is enabled.

8.4.6.6 Locate HSM boot header (state 44)

The SSCM is responsible for locating the HSM boot header. The SSCM reads several memory locations in flash memory to determine whether a valid HSM boot header is present. Specifically, the SSCM reads locations 0x60_0000, 0x60_4000 and 0x61_4000. A valid HSM boot header consists of three words: an HSM_Boot_Header_ID, a Start Address for the HSM application code, and a third word which is reserved for future use. By definition, a valid HSM_Boot_Header_ID is 0xFFFF_0000_FFFF_0000.

8.4.6.7 Valid Hardware Security Module header file? (state D)

In the present scenario, no valid HSM boot header is present. In this case, the SSCM takes the same path as if the HSM was not enabled at decision point C.

8.4.6.8 Provide reset vectors to IOP (state 21)

During PHASE3[DEST] of the Reset Generation Module's sequence, the start address for execution of the BAF code in the Boot Assist Flash is transferred from the DCF record to the MC_ME.

The SSCM releases reset to the IOP and then turns on the clock to the IOP. The clock to the HSM is turned off in this state even though the HSM was originally enabled. Because no valid header file for the HSM is found, the IOP goes to state 21 from decision point D, which turns the HSM clock off. The start address is transferred to the IOP at this time under the control of the SSCM and the IOP begins execution of the Boot Assist Flash code.

8.4.6.9 IOP begins BAF code execution (state 22)

After the reset signal is released and the respective clocks are turned on to the IOP, the IOP begins execution of code in the Boot Assist Flash.

8.4.6.10 IOP continues BAF code execution (state 23)

The BAF code begins by initializing registers and applying device settings. The BAF code is written by the device manufacturer and changes from device to device.

8.4.6.11 Calibration sequence (state 24)

This state is used to detect the presence of an emulation and debug device (ED) normally used in place of a regular production SPC57EM80xx (PO). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The ED consists of a PO and a special Buddy Device (BD) in a single package.

The IOP executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is present, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of SPC57EM80xx reset.

In this example, it is assumed that an ED is not being used in place of a regular SPC57EM80xx production device.

8.4.6.12 IOP checks to determine if HSM is enabled (state E)

The IOP, executing BAF code, checks to see if the HSM is enabled. In the present case, the SSCM has already disabled the clock to the HSM. Therefore, the HSM is not enabled.

8.4.6.13 IOP searches for boot header (state 31)

The IOP continues code execution by searching for the boot header. In the present example, it is assumed that no boot header file is present.

This is the header file for the CPU that starts executing the user application code. Generally, the BOOT CPU is either Core_0 or Core_1.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The IOP searches for the boot header, which can be located at one of eight addresses:

- 0xFC_0000
- 0xFC_4000
- 0xFC_8000
- 0xFC_C000
- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of seven words:

- Boot Header_ID/Boot_CPU
- Start address
- Configuration bits
- Configuration bits
- CPU_0 reset vector
- CPU_1 reset vector
- Checker Core_0s reset vector

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

Only four bits in the 16-bit Boot_CPU field are used. Bit 0 selects the I/O Processor, bit 1 selects CPU_0, bit 2 selects CPU_1, and bit 3 selects the Checker Core_0s. Bits 4–15 are not implemented.

8.4.6.14 Valid boot header is not found (decision point N)

The IOP, executing code in the Boot Assist Flash, does not find a valid boot header file in flash memory. Because there is no valid boot header, the serial boot mode must be entered to obtain an application program.

Note: This mode receives a program to be executed over a serial port implemented using the LINFlexD port. Once the program is received, using a defined format, the IOP jumps to the start address of the program and begins execution.

8.4.6.15 Initialize RAM (state 35)

The IOP, executing code in the Boot Assist Flash, initializes RAM areas in preparation for entering the serial boot mode. The serial boot mode configures the LINFlexD port to receive code from an external source. The LINFlexD port comes in a number of hardware configurations. The SPC57EM80xx uses LINFlexD, which implements a basic UART. Code in the BAF operates the LINFlexD module and implements the protocol.

8.4.6.16 Enter serial boot mode (state 14)

[Figure 43 \(Boot-up sequence, part E\)](#) shows the boot loader flow.

8.4.6.17 Life Cycle check indicates FIELD (decision point G)

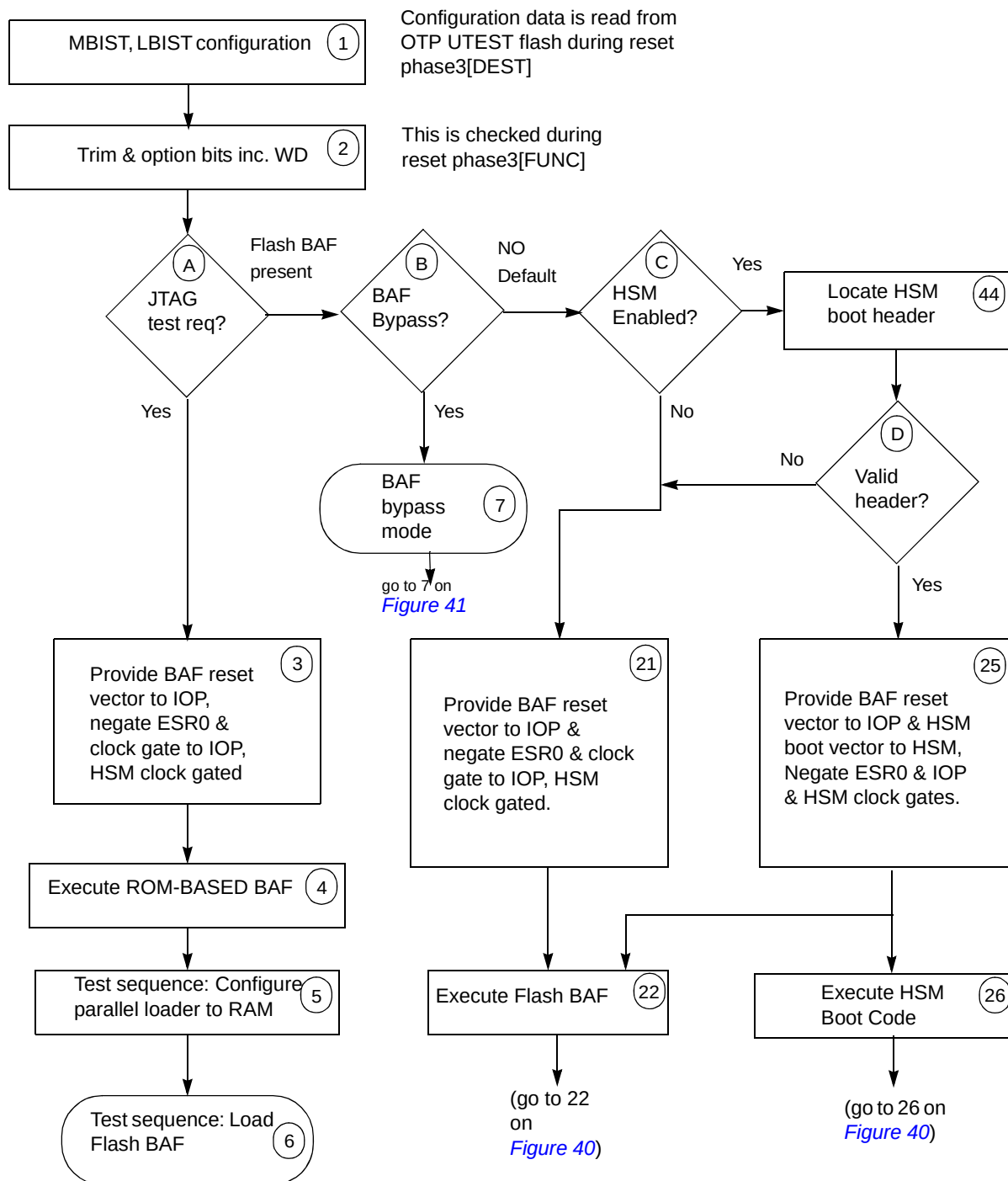
There are two possible settings for the Life Cycle parameter—FIELD and FACTORY. A setting of FIELD indicates that application code has been programmed into the flash memory of the device. In this case, the IOP enters a “wait mode” and remain there until a watchdog reset occurs.

8.4.6.18 Stop—wait for watchdog (state 36)

The present example would only be encountered when the SPC57EM80xx's program flash memory is incorrectly programmed.

Figure 39. Boot-up sequence part A

LOGIC / SSCM



Boot Flow and interaction with other modules (part 1)

Figure 40. Boot-up sequence part B

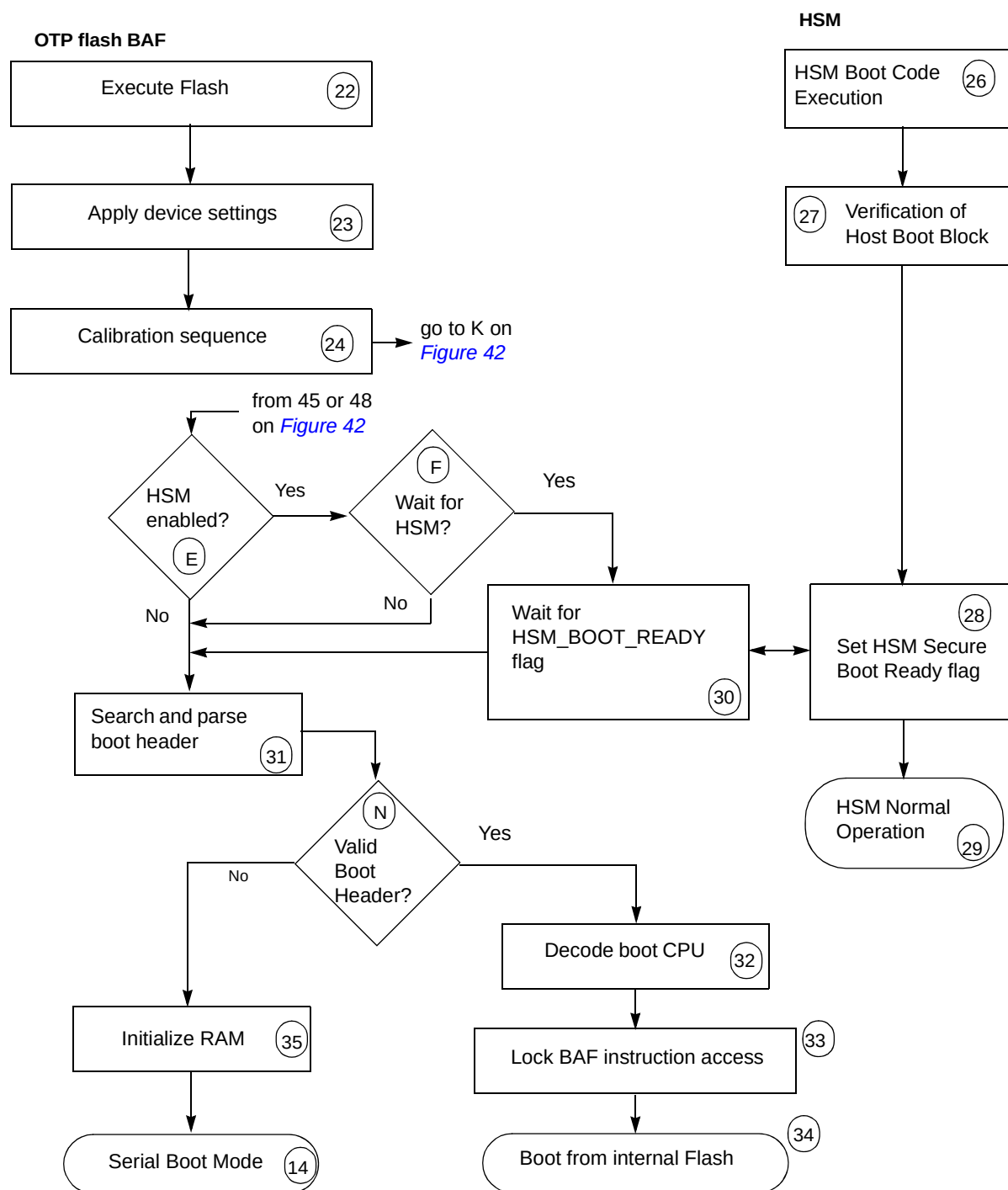


Figure 41. Boot-up sequence, part C

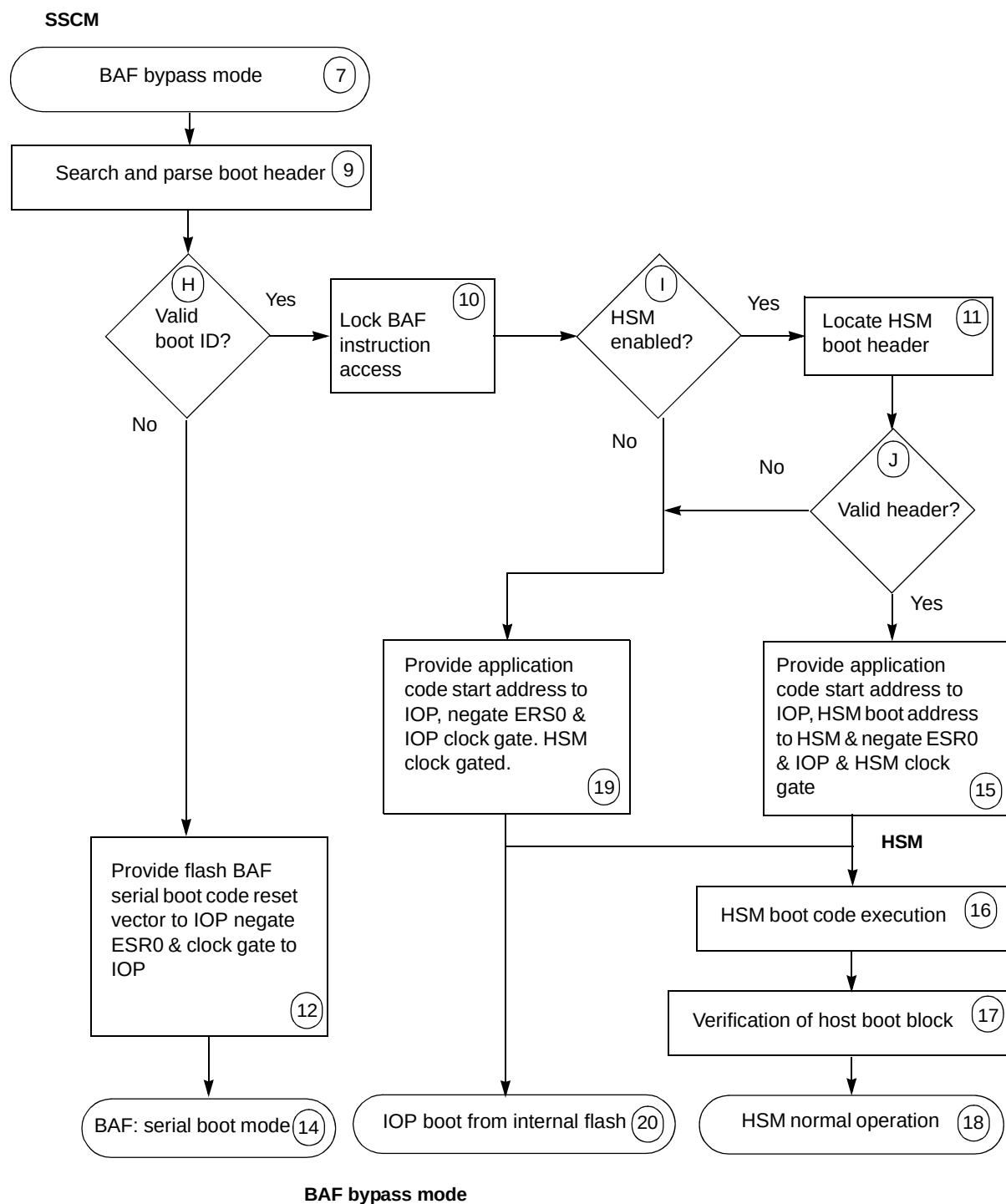


Figure 42. Boot-up sequence, part D

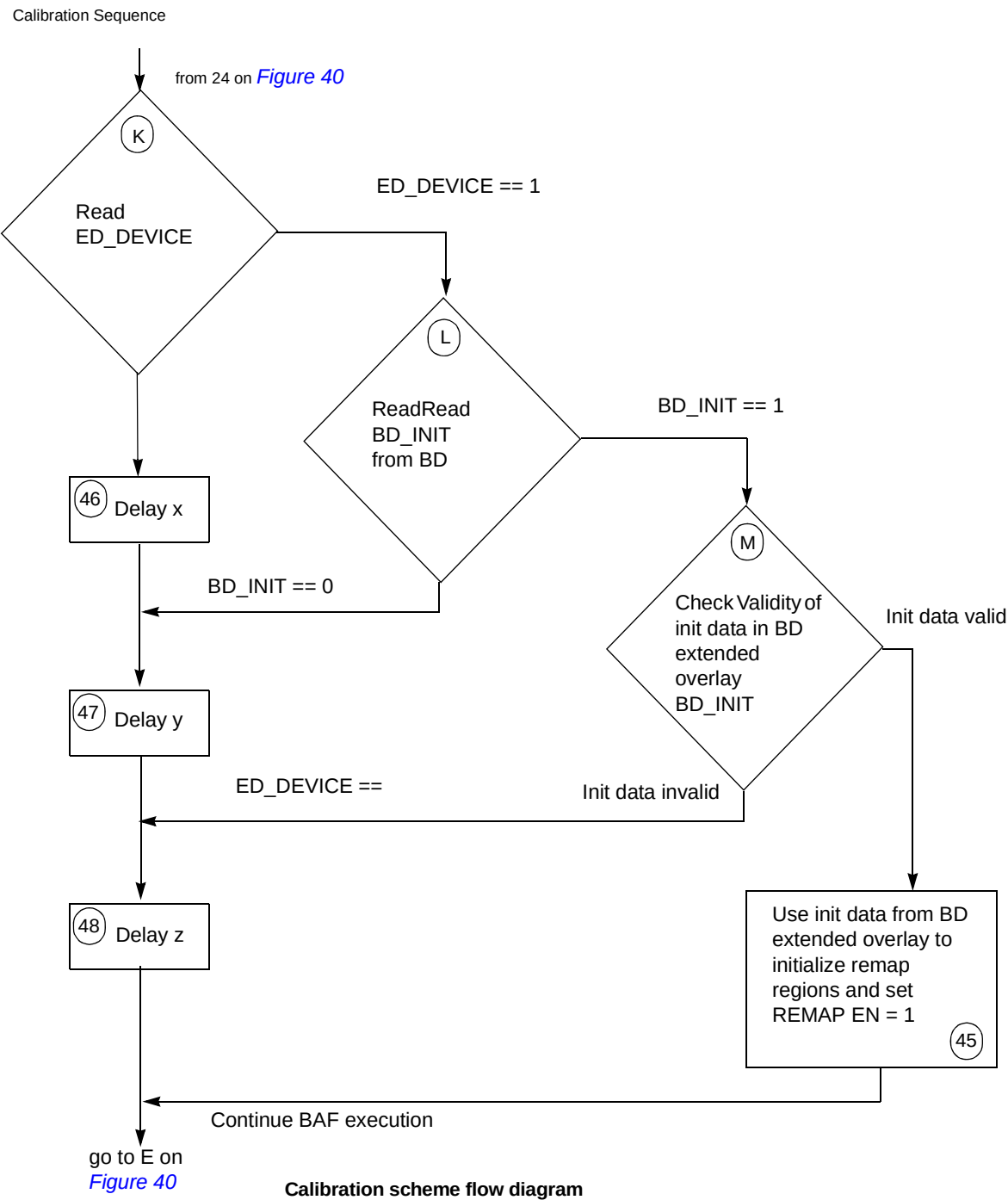
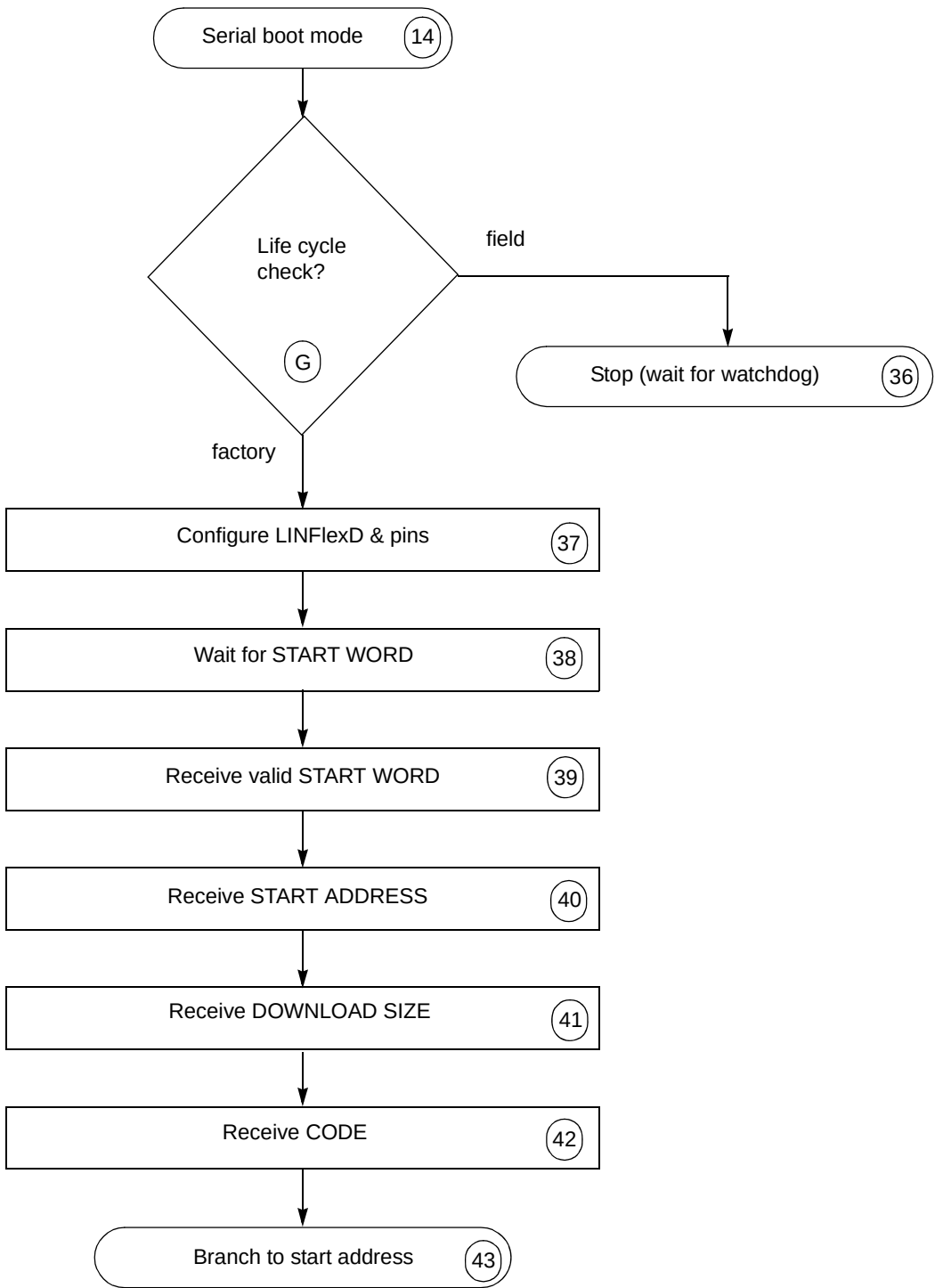


Figure 43. Boot-up sequence, part E



Serial boot process flow diagram

9 Device Configuration Format (DCF) Records

9.1 Introduction

Device Configuration Format (DCF) records are used to configure certain registers in the device during system boot while the reset signal is asserted. An individual DCF record consists of a pointer to the location of a register internal to the device and the data to be written to that register.

There are two broad categories of DCF records, TEST DCF Records and UTEST DCF Records. TEST DCF records are developed by the factory and are used mainly to program registers involved in trimming trip points for voltage comparators, adjusting analog to digital voltage supplies, trimming oscillator frequencies, and enabling RAM repair. However these records can also be used to write any DCF client. The TEST DCF Records are programmed into TEST flash during production and cannot be modified. TEST flash is not visible to the user.

The second set of DCF records are UTEST DCF records. Some UTEST DCF records are written by the factory and programmed during production testing (see [Table 16](#) in [Chapter 5: Memory Map](#)). Others are written by the end user and programmed at the same time application code is programmed into the flash memory. User-supplied UTEST DCF records start at the next location in the UTEST memory map following the UTEST DCF records programmed into the flash memory during factory production. The user-defined DCF records set up the initial memory map, define which tests the Self-Test Control Unit (STCU2) runs during the boot sequence, enable the HSM, assign memory blocks as OTP, assign flash blocks to be associated with specific password groups and assign flash memory blocks to specific tamper detect regions.

System boot is a complex process requiring a considerable amount of initialization to take place before releasing reset. Before the device can be used in an application, the user application code, reset vectors for all of the CPUs and the DCF records must be properly programmed into their respective flash memories.

When power is applied to a properly programmed device, the Power Management Controller (PMC) takes control of the device. After the system power supplies have reached predefined levels, the Power Management Controller signals the Reset Generation Module (RGM) to begin the boot sequence. During the boot sequence, the RGM enables another internal module, the System Status and Control Module (SSCM), to read the device configuration records and then write the configuration information to the specified registers. After registers are initialized, the SSCM starts the (STCU2), which begins running the tasks enabled by the DCF record information.

9.2 DCF clients

DCF clients are 32-bit wide hardware registers inside a module, that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features. DCF Clients have a default value before any DCF Records are written, and may have special writing constraints, such as, 'Write Once' or allowing bits to be written from '1' to '0' only or vice versa. DCF clients need not implement all 32 bits. DCF clients maybe designated 'TEST DCF record only', which means that only DCF Records stored in TEST flash can write to the DCF Client.

A list of DCF clients is shown in [Table 112](#). (TEST DCF Record Only Clients are not shown as these are not accessible by the user).

The DCF Records select the 'target' DCF client by a 30-bit field in the DCF Record consisting of a 15-bit 'Chip Select' field and a 15-bit Address field. Each module that includes DCF clients is assigned a Chip Select during chip definition. The address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

9.3 DCF records

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST flash memory beginning at 0x0040_0200.

The structure of a DCF Record is shown in [Figure 44](#).

Figure 44. DCF record structure

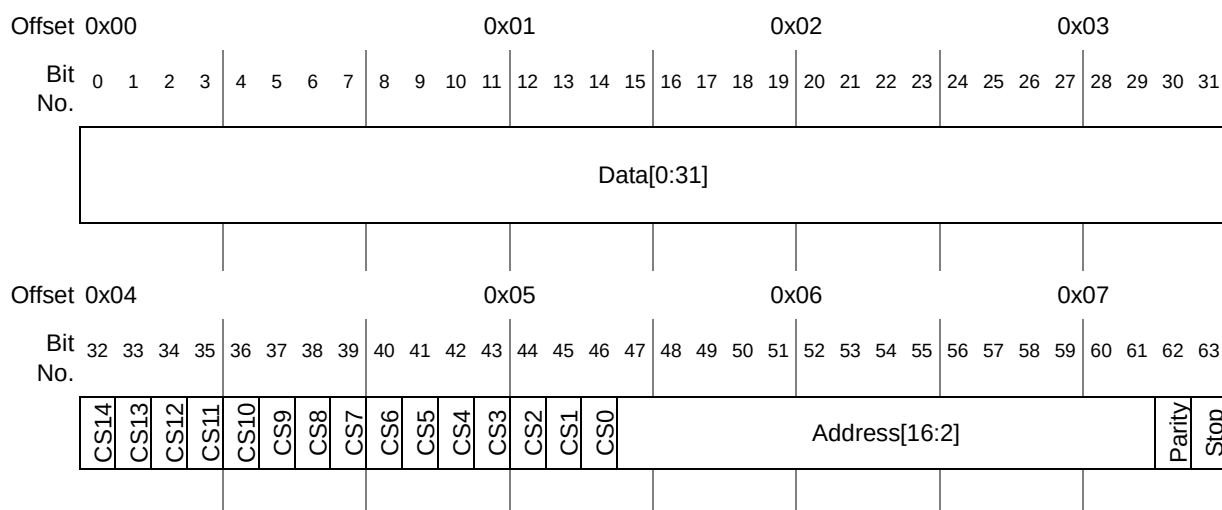


Table 110. DCF record field description

Field	Description
0–31 Data[0:31]	32 bits of data that is to be written to the DCF Client
32–46 CSn	Chip Select n One Chip select is asserted ('1') per DCF record to select the target module for the DCF client. All other Chip Selects should be negated ('0')
47–61 Address[16:2]	Address of the DCF client within the selected module Note: Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF Client addresses are defined in each module chapter

Table 110. DCF record field description

Field	Description
62 Parity	Parity Bit for the DCF Record Note: This bit is NOT implemented for DCF Client written from UTEST
63 Stop	Stop bit This bit indicates the end the list of DCF Records. Stop = '0' - NOT the end of the list Stop = '1' - End of the list Note: The Erased state of flash is 0xFFFF_FFFF_FFFF_FFFF. Therefore the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list

The following structure must be present:

- The first record must be a start record.
- DCF records containing configuration data must immediately follow the start record with no blank records between—an unprogrammed record is interpreted as a stop record and no DCF records following that record are processed.
- The end of the configuration records are indicated by the presence of a stop record.

Each record is 64 bits in length. [Figure 45](#) shows the DCF start record.

Warning: The start record must be placed at the beginning of the DCF area in UTEST flash memory to indicate to the device that the following data records must be processed.

Figure 45. DCF start record

0x00 0:31	0x04 32:63
0x05AA55AF	0x00000000

The stop bit designates the last device configuration record in a set of records. Normally, the stop bit is never set in a valid UTEST DCF record programmed during production, since this would prevent additional UTEST records from being appended to the factory programmed UTEST DCF records. However, the flash memory location following the last UTEST DCF record programmed at the factory is an unprogrammed location. Therefore, that location's contents is 0xFFFF_FFFF_FFFF_FFFF. Thus, the stop bit location in this unprogrammed flash memory location is a logic 1, signifying that this is the last DCF record and it is not to be acted upon.

During the boot sequence, the SSCM uses the DCF records to program various registers before reset is released. Some device configuration records are calculated during production testing and programmed into the TEST flash memory area. Other DCF records are developed by the end user and programmed into the UTEST flash area. A device configuration record simply consists of an internal register address and a data field. During system boot, while the reset signal is asserted, the SSCM reads the device configuration



records and writes the data to the appropriate internal module registers. Thus, many operational aspects of the device are properly configured when the reset signal is released and normal device operation begins.

The stop record indicates the end of DCF records to be processed. The general format of the stop record is shown in [Figure 46](#). Only the stop bit needs to be 1 in order to form a stop record—all other bits are ignored. An unprogrammed location in UTEST flash is interpreted as a stop record.

Figure 46. DCF stop record

0:31	32:62	63
Reserved	Reserved	1

If n data records are to be stored in UTEST, the data structure must be as shown in [Table 111](#).

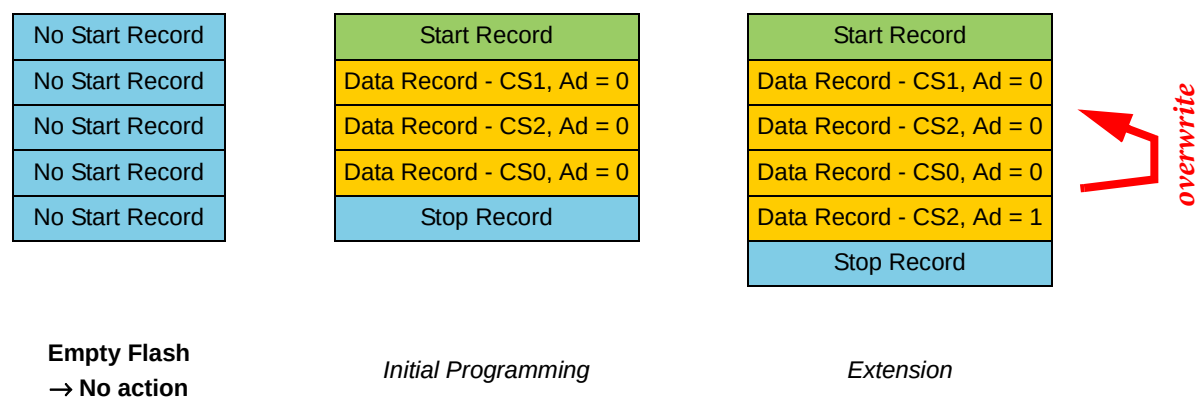
Table 111. Series of DCF records in UTEST flash memory

ADDR offset	DATA			
0x00	0x05AA55AF			
0x04	0x00000000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
$8n-1 + 0x0$	Reserved			
$8n-1 + 0x4$	Reserved			1
$8n + 0x0$				
$8n + 0x4$				

Warning: There must never be an unprogrammed record in the DCF data structure, as it is interpreted as a stop record and subsequent records are ignored. This allows one to program the records in several sessions, each time appending new records at the end of the list, as shown in [Figure 47](#).

It is possible for more than one DCF Record to write to the same DCF client. In this case, as shown in the “Extension” part of [Figure 47 \(Appending DCF records\)](#), the later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

Figure 47. Appending DCF records



9.3.1 UTEST DCF records

UTEST DCF records are located in the UTEST flash memory area (refer to UTEST flash memory map in the Memory map chapter). Some UTEST DCF records are programmed at the factory. The user may add additional USER DCF records beginning with the first unprogrammed location following the factory programmed UTEST DCF records. An unprogrammed location in flash memory contains 0xFFFF_FFFF.

When programming UTEST DCF records at the factory, the records must start at the first address in the UTEST flash memory area and be continuous; that is, one UTEST DCF record must immediately follow the previous UTEST DCF record. An unprogrammed location in the UTEST flash memory area is interpreted by the SSCM as the end of the UTEST DCF records. When this happens, the SSCM passes control of the boot sequence to the Reset Generation Module, which directs the STCU to begin memory built in self test (MBIST) and logic built in self test (LBIST) execution.

9.4 DCF client table

A list of DCF clients is shown in [Table 112](#). Various attributes of the DCF clients are listed in these tables. The following list explains the meaning of the columns and give a description of the attributes of these DCF clients.

- **DCF Chip Select**
The CS field is a 15-bit field with each bit in the field assigned to a particular module. These bits act as module select signals for the internal modules. Only one bit in this field should be set in a DCF record.
- **DCF address**
The ADDR field is a 15-bit field that specifies an internal address for a DCF client. These addresses are used only by the SSCM when interpreting DCF records and writing the data to specified modules.
- **DCF Client Description**
This field contains the name of the DCF clients.
- **Reset value of DCF client**
This field contains the reset value of DCF client before being written by the SSCM with the associated DCF record.
- **Typical value of DCF client**
This field contains a typical value that you might program into the various DCF clients. This field is for information only.
- **IPS read**
This column specifies whether or not a DCF client can be read by user software after the release of reset.
- **DCF client special strategy**
Though all DCF records have the same format, they can be used in different ways. As the device is being designed, registers must be designated as Device Configuration Format clients if they need to be written with a DCF record. There are other features that must be designated for DCF clients. For instance, DCF clients can have several different write strategies—Parity, Write Once, Triple Voted + Parity + Check with second write, Triple voted + Check with second write, or no strategy at all. The SSCM examines the DCF client to determine the correct special strategy to use and then appropriately write a DCF record to that client. The following list defines the special strategies.
 - **None**
No special DCF strategy is used.
 - **Parity**
Not implemented for DCF clients. Only used for TEST only DCF clients not accessible by the user.
 - **Write Once**
A register using the Write Once strategy can only be written once. The DCF client ignores subsequent writes.
 - **Triple Voted**
DCF clients that use the Triple Voted strategy have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value.

Triple voting allows for a 'bit-flip' error to occur without changing the DCF client output data.

- Triple Voted with second write

DCF clients that use the Triple Voted with 2nd write strategy have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value. During the second execution of Phase 3 of the reset sequence, the SSCM attempts to write the DCF client again. At this time, the DCF client checks to see that the register contains the same data that is being written again.

- Write 0 only

A bit in a DCF client can only be written from a logic 1 to a logic 0. An attempt to write a bit with this attribute to a logic 1 is ignored.

- Write 1 only

A bit in a DCF client can only be written from a logic 0 to a logic 1. An attempt to write a bit with this attribute to a logic 0 is ignored.

- DCF order dependency in flash memory

This column details whether or not there are special conditions on the location of a particular DCF record in flash memory.

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
STCU - Support for 10 LBIST partitions; 69 Memory Cuts; 32-bit PRPG; 64-bit MISR									
000_0000_0000_0100	0000000000000010	STCU_SKC	—	—	yes	None	None	First Record among STCU, Key1/Key2 dcf record sequence	16
000_0000_0000_0100	0000000000000000	STCU_RUN	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000000011	STCU_CFG	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000000100	STCU_PLL_CFG	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000000101	STCU_WDG	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000000111	STCU_CRCE	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000001010	STCU_ERR_FM	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000001111	STCU_LBRMSW	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000010000	STCU_LBCFM	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000011101	STCU_MBCFML[31:0]	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000011110	STCU_MBCFMM[31:0]	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000000011111	STCU_MBCFMH[31:0]	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000001000000	STCU_LB_CTRL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000001000001	STCU_LB_PCS_0	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000001000010	STCU_LB_PRPGL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	0000000001000011	STCU_LB_PRPGH_0	—	—	yes	None	None	None	0

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_0100	000000001000100	STCU_LB_MISREL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001000101	STCU_LB_MISREH_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001001000	STCU_LB_MISRELSW_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001001001	STCU_LB_MISREHSW_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010000	STCU_LB_CTRL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010001	STCU_LB_PCS_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010010	STCU_LB_PRPGL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010011	STCU_LB_PRPGH_1	—	—	yes	None	None	None	0
000_0000_0000_0100	000000001010100	STCU_LB_MISREL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010101	STCU_LB_MISREH_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001011000	STCU_LB_MISRELSW_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001011001	STCU_LB_MISREHSW_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100000	STCU_LB_CTRL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100001	STCU_LB_PCS_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100010	STCU_LB_PRPGL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100011	STCU_LB_PRPGH_2	—	—	yes	None	None	None	0
000_0000_0000_0100	000000001100100	STCU_LB_MISREL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100101	STCU_LB_MISREH_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001101000	STCU_LB_MISRELSW_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001101001	STCU_LB_MISREHSW_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001110000	STCU_LB_CTRL_3	—	—	yes	None	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_0100	000000001110001	STCU_LB_PCS_3	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001110010	STCU_LB_PRPGL_3	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001110011	STCU_LB_PRPGH_3	—	—	yes	None	None	None	0
000_0000_0000_0100	000000001110100	STCU_LB_MISREL_3	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001110101	STCU_LB_MISREH_3	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001111000	STCU_LB_MISRELSW_3	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001111001	STCU_LB_MISREHSW_3	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010000000	STCU_LB_CTRL_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010000001	STCU_LB_PCS_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010000010	STCU_LB_PRPGL_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010000011	STCU_LB_PRPGH_4	—	—	yes	None	None	None	0
000_0000_0000_0100	000000010000100	STCU_LB_MISREL_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010000101	STCU_LB_MISREH_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010001000	STCU_LB_MISRELSW_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010001001	STCU_LB_MISREHSW_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010010000	STCU_LB_CTRL_5	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010010001	STCU_LB_PCS_5	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010010010	STCU_LB_PRPGL_5	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010010011	STCU_LB_PRPGH_5	—	—	yes	None	None	None	0
000_0000_0000_0100	000000010010100	STCU_LB_MISREL_5	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010010101	STCU_LB_MISREH_5	—	—	yes	None	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_0100	000000010011000	STCU_LB_MISRELSW_5	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010011001	STCU_LB_MISREHSW_5	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010100000	STCU_LB_CTRL_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010100001	STCU_LB_PCS_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010100010	STCU_LB_PRPGL_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010100011	STCU_LB_PRPGH_6	—	—	yes	None	None	None	0
000_0000_0000_0100	000000010100100	STCU_LB_MISREL_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010100101	STCU_LB_MISREH_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010101000	STCU_LB_MISRELSW_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010101001	STCU_LB_MISREHSW_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010110000	STCU_LB_CTRL_7	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010110001	STCU_LB_PCS_7	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010110010	STCU_LB_PRPGL_7	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010110011	STCU_LB_PRPGH_7	—	—	yes	None	None	None	0
000_0000_0000_0100	000000010110100	STCU_LB_MISREL_7	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010110101	STCU_LB_MISREH_7	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010111000	STCU_LB_MISRELSW_7	—	—	yes	None	None	None	8
000_0000_0000_0100	000000010111001	STCU_LB_MISREHSW_7	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011000000	STCU_LB_CTRL_8	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011000001	STCU_LB_PCS_8	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011000010	STCU_LB_PRPGL_8	—	—	yes	None	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_0100	000000011000011	STCU_LB_PRPGH_8	—	—	yes	None	None	None	0
000_0000_0000_0100	000000011000100	STCU_LB_MISREL_8	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011000101	STCU_LB_MISREH_8	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011001000	STCU_LB_MISRELSW_8	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011001001	STCU_LB_MISREHSW_8	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011010000	STCU_LB_CTRL_9	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011010001	STCU_LB_PCS_9	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011010010	STCU_LB_PRPGL_9	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011010011	STCU_LB_PRPGH_9	—	—	yes	None	None	None	0
000_0000_0000_0100	000000011010100	STCU_LB_MISREL_9	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011010101	STCU_LB_MISREH_9	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011011000	STCU_LB_MISRELSW_9	—	—	yes	None	None	None	8
000_0000_0000_0100	000000011011001	STCU_LB_MISREHSW_9	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000000	STCU_MB_CTRL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000001	STCU_MB_CTRL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000010	STCU_MB_CTRL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000011	STCU_MB_CTRL_3	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000100	STCU_MB_CTRL_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000101	STCU_MB_CTRL_5	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000110	STCU_MB_CTRL_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000111	STCU_MB_CTRL_7	—	—	yes	None	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_0100	000000110001000	STCU_MB_CTRL_8	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001001	STCU_MB_CTRL_9	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001010	STCU_MB_CTRL_10	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001011	STCU_MB_CTRL_11	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001100	STCU_MB_CTRL_12	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001101	STCU_MB_CTRL_13	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001110	STCU_MB_CTRL_14	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001111	STCU_MB_CTRL_15	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110010000	STCU_MB_CTRL_16	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110010001	STCU_MB_CTRL_17	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110010010	STCU_MB_CTRL_18	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110010011	STCU_MB_CTRL_19	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110010100	STCU_MB_CTRL_20	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110010101	STCU_MB_CTRL_21	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110010110	STCU_MB_CTRL_22	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110010111	STCU_MB_CTRL_23	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110011000	STCU_MB_CTRL_24	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110011001	STCU_MB_CTRL_25	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110011010	STCU_MB_CTRL_26	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110011011	STCU_MB_CTRL_27	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110011100	STCU_MB_CTRL_28	—	—	yes	None	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_0100	000000110011101	STCU_MB_CTRL_29	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110011110	STCU_MB_CTRL_30	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110011111	STCU_MB_CTRL_31	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110100000	STCU_MB_CTRL_32	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110100001	STCU_MB_CTRL_33	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110100010	STCU_MB_CTRL_34	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110100011	STCU_MB_CTRL_35	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110100100	STCU_MB_CTRL_36	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110100101	STCU_MB_CTRL_37	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110100110	STCU_MB_CTRL_38	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110100111	STCU_MB_CTRL_39	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110101000	STCU_MB_CTRL_40	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110101001	STCU_MB_CTRL_41	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110101010	STCU_MB_CTRL_42	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110101011	STCU_MB_CTRL_43	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110101100	STCU_MB_CTRL_44	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110101101	STCU_MB_CTRL_45	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110101110	STCU_MB_CTRL_46	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110101111	STCU_MB_CTRL_47	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110110000	STCU_MB_CTRL_48	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110110001	STCU_MB_CTRL_49	—	—	yes	None	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_0100	000000110110010	STCU_MB_CTRL_50	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110110011	STCU_MB_CTRL_51	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110110100	STCU_MB_CTRL_52	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110110101	STCU_MB_CTRL_53	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110110110	STCU_MB_CTRL_54	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110110111	STCU_MB_CTRL_55	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110111000	STCU_MB_CTRL_56	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110111001	STCU_MB_CTRL_57	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110111010	STCU_MB_CTRL_58	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110111011	STCU_MB_CTRL_59	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110111100	STCU_MB_CTRL_60	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110111101	STCU_MB_CTRL_61	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110111110	STCU_MB_CTRL_62	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110111111	STCU_MB_CTRL_63	—	—	yes	None	None	None	8
000_0000_0000_0100	000000111000000	STCU_MB_CTRL_64	—	—	yes	None	None	None	8
000_0000_0000_0100	000000111000001	STCU_MB_CTRL_65	—	—	yes	None	None	None	8
000_0000_0000_0100	000000111000010	STCU_MB_CTRL_66	—	—	yes	None	None	None	8
000_0000_0000_0100	000000111000011	STCU_MB_CTRL_67	—	—	yes	None	None	None	8
000_0000_0000_0100	000000111000100	STCU_MB_CTRL_68	—	—	yes	None	None	None	8
000_0000_0000_0100	000000111000101	STCU_MB_CTRL_69	—	—	yes	None	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
PASS									
000_0000_0000_1000	000000000101000	Life Cycle (LCSTAT)	0xff	—	yes	Write '0' Only	None	None	8
000_0000_0000_1000	000000000101001 – 000000000101011	Reserved							
000_0000_0000_1000	000000000101100	Censorship	0x0000	0x55AA	no	None	None	None	8
000_0000_0000_1000	000000000101101 – 000000000101111	Reserved							
000_0000_0000_1000	000000000110000	Production Disable	0x0000	—	no	Write Once	None	None	8
000_0000_0000_1000	000000000110001 – 000000000111111	Reserved							
000_0000_0000_1000	000000001000000	LOCK0_PG0	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001000001	LOCK1_PG0	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001000010	LOCK2_PG0	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001000011	LOCK3_PG0	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001000100	LOCK0_PG1	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001000101	LOCK1_PG1	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001000110	LOCK2_PG1	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001000111	LOCK3_PG1	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001001000	LOCK0_PG2	0xfff_ffff	—	yes	None	None	None	8
000_0000_0000_1000	000000001001001	LOCK1_PG2	0xfff_ffff	—	yes	None	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_1000	000000001001010	LOCK2_PG2	0xfff_fff	—	yes	None	None	None	8
000_0000_0000_1000	000000001001011	LOCK3_PG2	0xfff_fff	—	yes	None	None	None	8
000_0000_0000_1000	000000001001100	LOCK0_PG3	0xfff_fff	—	yes	None	None	None	8
000_0000_0000_1000	000000001001101	LOCK1_PG3	0xfff_fff	—	yes	None	None	None	8
000_0000_0000_1000	000000001001110	LOCK2_PG3	0xfff_fff	—	yes	None	None	None	8
000_0000_0000_1000	000000001001111	LOCK3_PG3	0xfff_fff	—	yes	None	None	None	8
000_0000_0000_1000	000000001001100 – 0000000011111111	Reserved							
Tamper Detect									
000_0000_0001_0000	0000000000000000	Diary Base Address	0xFFFF + FFFC	—	no	Write Once + Write '0' only	None	None	8
000_0000_0001_0000	0000000000000001	Tamper Region Override	0x00	—	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000000000010 – 0000000000000011	Reserved							
000_0000_0001_0000	0000000000001000	OTP_EN0	0x0000 _0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000000001001	OTP_EN1	0x0000 _0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000000001010	OTP_EN2	0x0000 _0000	—	no	Write '1' only	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0001_0000	0000000000001011	OTP_EN3	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000000001100 – 00000000000010011	Reserved							
000_0000_0001_0000	000_0000_0001_0100	TDR0_LOCK0	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_0101	TDR0_LOCK1	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_0110	TDR0_LOCK2	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_0111	TDR0_LOCK3	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_1000	TDR1_LOCK0	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_1001	TDR1_LOCK1	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_1010	TDR1_LOCK2	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_1011	TDR1_LOCK3	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_1100	TDR2_LOCK0	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_1101	TDR2_LOCK1	0x0000_0000	—	no	Write '1' only	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0001_0000	000_0000_0001_1110	TDR2_LOCK2	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0001_1111	TDR2_LOCK3	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_0000	TDR3_LOCK0	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_0001	TDR3_LOCK1	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_0010	TDR3_LOCK2	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_0011	TDR3_LOCK3	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_0100	TDR4_LOCK0	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_0101	TDR4_LOCK1	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_0110	TDR4_LOCK2	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_0111	TDR4_LOCK3	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_1000	TDR5_LOCK0	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_1001	TDR5_LOCK1	0x0000_0000	—	no	Write '1' only	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0001_0000	000_0000_0010_1010	TDR5_LOCK2	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000_0000_0010_1011	TDR5_LOCK3	0x0000_0000	—	no	Write '1' only	None	None	8
000_0000_0001_0000	000000000101100 – 000000011111111	Reserved							
HSM									
000_0000_0010_0000	000000000010000	HSM Enable and Configuration	0x0000 0000		no	wr_once	None	None	8
000_0000_0010_0000	000000000010001 – 000000000011111	Reserved							
000_0000_0010_0000	000000000010000	HSM Exclusive Flags	0x0000 0000		no	wr_once	None	None	8
000_0000_0010_0000	000000000010001 – 000000000011111	Reserved							
000_0000_0010_0000	000000000100000	HSM Alternate Prog Interface Flags	0x0000 0000	—	no	wr_once	None	None	8
000_0000_0010_0000	000000000100001 – 000000011111111	Reserved							
MISCELLANEOUS									
000_0000_1000_0000	000000000000000	UTEST Miscellaneous (See <i>Figure 48</i> and <i>Table 113</i> for bit position details.)	0x0000 0941	—	no	Triple Voted	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_1000_0000	0000000000000001 – 0000000000011110	Reserved							
000_0000_1000_0000	0000000000011111	PMC_REE bits	0x0000 0001	—	no	Triple Voted	Non- Differenti al	None	8
000_0000_1000_0000	000000000100000 – 000000000110101	Reserved							
000_0000_1000_0000	000000000110110	VREG1P2_ENB (VREG1P2_ENB is set during a destructive reset. This will turn off the Auxiliary Regulator.)	0x0000 0001	—	no	Triple Voted	Non- Differenti al	None	8
000_0000_1000_0000	000000000110111 – 000000011111111	Reserved							
000_0000_1000_0000	0000000000000001 – 000000011111111	Reserved							
BAF 'Soft' Clients									
100_0000_0000_0000	0x0000	Security Watchdog Control Register	NA		no	Write Once	None	None	8
100_0000_0000_0000	0x0001	Security Watchdog Timeout	NA		no	Write Once	None	None	8
100_0000_0000_0000	0x0002	Security Watchdog Service Address	NA		no	Write Once	None	None	8

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
100_0000_0000_0000	0x0003	Security Watchdog CPU select	NA		no	Write Once	None	None	8
100_0000_0000_0000	0x0004 – 0x01FF	Reserved	NA						
100_0000_0000_0000	0x0200	Address	NA		no	None	None	Must be written first as a pair with a data register	8
100_0000_0000_0000	0x0201	32-bit Data	NA		no	None	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	0x0202	16-bit Data	NA		no	None	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	0x0203	8-bit Data	NA		no	None	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	0x0204 – 0x020F	Reserved	NA						

Table 112. DCF client list⁽¹⁾

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IP read direct access	DCF client special strategy	Flash special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
100_0000_0000_0000	0x0210	Callback address	NA		no	None	None	None	8
100_0000_0000_0000	0x0211 – 0x7FFF	Reserved	NA						

1. DCF programming should start from address 0x00400208.

Figure 48. UTEST Miscellaneous DCF Client

Offset: 0000000000000000

Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	Reserved	XOSC ALC_DIS	XOSC LF_EN	RMII MODE	3p3V_5V	OSC EN_40MHZ	Reserved	LOCKSTEP EN	XOSC EN	XOSC EXT CLOAD	XOSC LOAD CAP SEL[4:0]					ESR0 GATE

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 113. UTEST Miscellaneous DCF Client field descriptions

Field	Description
0-16	Reserved
17 XOSC ALC_DIS	XOSC ALC_DIS 0 Enable Automatic Level Controller 1 Disable Automatic Level Controller
18 XOSC LF_EN	XOSC LF_EN 0 Oscillator is not used for low frequency (~8 MHz) 1 Oscillator is used for low frequency (~8 MHz)
19 RMII MODE	RMII MODE — RMII Mode Enable 0 FEC configured for MII Mode 1 FEC configured for RMII Mode
20 3p3V_5V	3p3_5V — FlexRay 3.3 V vs. 5 V indicator 0 3.3 V 1 5 V
21 OSC EN_40MHZ	XOSC EN_40MHz 0 20 MHz 1 40 MHz XOSC
22	Reserved
23 LOCKSTEP EN	LOCKSTEP EN — Enable Safety Core to be in lockstep with Checker Core 0 Lockstep disabled 1 Lockstep enabled

Table 113. UTEST Miscellaneous DCF Client field descriptions (continued)

Field	Description
24 XOSC EN	XOSC EN — Enable XOSC 0 XOSC disabled 1 XOSC enabled
25 XOSC EXT CLOAD	XOSC EXT CLOAD 0 Selects XOSC internal cap (ok for 20–40 MHz operation) 1 External (required for 4–16 MHz operation)
26-30 XOSC LOAD CAP SEL[0:4]	XOSC LOAD CAP SEL [0:4] — Five trim bits to program the load capacitance. Load capacitor values are determined by the crystal manufacturer data sheet.
31 ESR0 GATE	ESR0 Gate — Gate ESR0 Input 0 ESR0 Input Disabled. ESR0 only asserted as an output from the device. 1 ESR0 Input Enabled. Assertion of the ESR0 pin causes a hard system reset.

Figure 49. PMC_REE bits DCF Client

Offset: 00000000001111

Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	LVD3_C_VEL	LVD3_F	LVD4_C	HVD7_C	LVD10_F	LVD13_IM	Reserved	LVD14_IM	LVD14_A	LVD15_C	LVD3_P	HVD15_A	HVD7_F	HVD8_C	LVD10_A	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 114. PMC_REE bits DCF client field descriptions

Field	Description
0–15	Reserved
16 LVD3_C	LVD3_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD3[LVD3_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

Table 114. PMC_REE bits DCF client field descriptions (continued)

Field	Description
17 LVD3_F	LVD3_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a system reset. The RES_VD3[LVD3_F] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset. 1 Enabled. LVD assertion on the supply of the flash causes system reset.
18 LVD4_C	LVD4_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD4[LVD4_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
19 HVD7_C	HVD7_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD7[HVD7_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.
20 LVD10_F	LVD10_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a system reset. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset. 1 Enabled. LVD assertion on the supply of the flash causes system reset.
21 LVD13_IM	LVD13_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates a system reset. The RES_VD13[LVD13_IM] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes system reset.
22	Reserved
23 LVD14_IM	LVD14_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O Main generates a system reset. The RES_VD14[LVD14_IM] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O Main does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O Main causes system reset.
24 LVD14_A	LVD14_A reset enable. This bit defines whether an LVD assertion on the supply of the ADC generates a system reset. The RES_VD14[LVD14_A] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the supply of the ADC causes system reset.
25 LVD15_C	LVD15_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD15[LVD15_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

Table 114. PMC_REE bits DCF client field descriptions (continued)

Field	Description
26 LVD3_P	LVD3_P reset enable. This bit defines whether an LVD assertion on the supply of the PLL generates a system reset. The RES_VD3[LVD3_P] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the PLL does not cause system reset. 1 Enabled. LVD assertion on the supply of the PLL causes system reset.
27 HVD15_A	HVD15_A reset enable. This bit defines whether an HVD assertion on the first supply of the ADC generates a system reset. The RES_VD15[HVD15_A] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the first supply of the ADC does not cause system reset. 1 Enabled. HVD assertion on the first supply of the ADC causes system reset.
28 HVD7_F	HVD7_F reset enable. This bit defines whether an HVD assertion on the supply of the flash generates a system reset. The RES_VD7[HVD7_F] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the flash does not cause system reset. 1 Enabled. HVD assertion on the supply of the flash causes system reset.
29 HVD8_C	HVD8_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD8[HVD8_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.
30 LVD10_A	LVD10_A reset enable. This bit defines whether an LVD assertion on the supply of the ADC generates a system reset. 0 Disabled. LVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the supply of the ADC causes system reset.
31	Reserved

10 Power management

10.1 Overview

SPC57EM80xx microcontrollers include a robust power management infrastructure that enables applications to select among various operational and low-power modes and to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can leave reset. This chapter gives an overview of the built-in power management features.

The power management infrastructure comprises the following modules:

- Power Control Unit (MC_PCU)
- Clock Generation Module (MC_CGM)
- Mode Entry Module (MC_ME)

This chapter describes these modules in brief. For complete details, see the respective module chapters.

The MC_CGM generates reference clocks for all the chip blocks. It works in conjunction with internal clock gating logic to implement low power modes.

The MC_PCU provides a status register and a submodule, the PMC_dig (Power Management Control Digital Interface), provides an interface to configure and control internal voltages.

The MC_ME module controls chip operational modes and mode transition sequences. It also contains configuration, control and status registers accessible for the application.

The SPC57EM80xx has two types of modes, SYSTEM modes and USER modes.

SYSTEM modes:

- SYSTEM modes are entered automatically on system events or on application request.
- The SYSTEM modes are: RESET, SAFE, TEST and DRUN.

Table 115. SYSTEM modes

Mode	Description	Entry	Exit
RESET	System remains in this mode until all resources are available for S/W to take device control. RESET manages H/W initialization of chip configuration, voltage regulators, oscillators, PLLs, and flash.	System reset assertion from RGM	System reset deassertion from RGM
SAFE	May be entered on detection of a recoverable error. Forces system into predefined safe configuration from which the system may try to recover.	H/W fail, S/W request from DRUN, TEST and RUNx	System reset assertion, DRUN through S/W
TEST	Control environment for device self-test. Allows application to run its own self-test like flash checksum and RAM BIST.	Software request from DRUN	System reset asserted, DRUN through S/W
DRUN	Entry mode for the embedded software. Provides full system accessibility and enables system configuration. Provides gate to enter USER modes. BAF, when present, is executed in DRUN mode.	Reset removed, from SAFE, TEST and RUNx through S/W, STANDBY wakeup	Reset applied, RUNx, TEST through S/W, SAFE through S/W or H/W fail

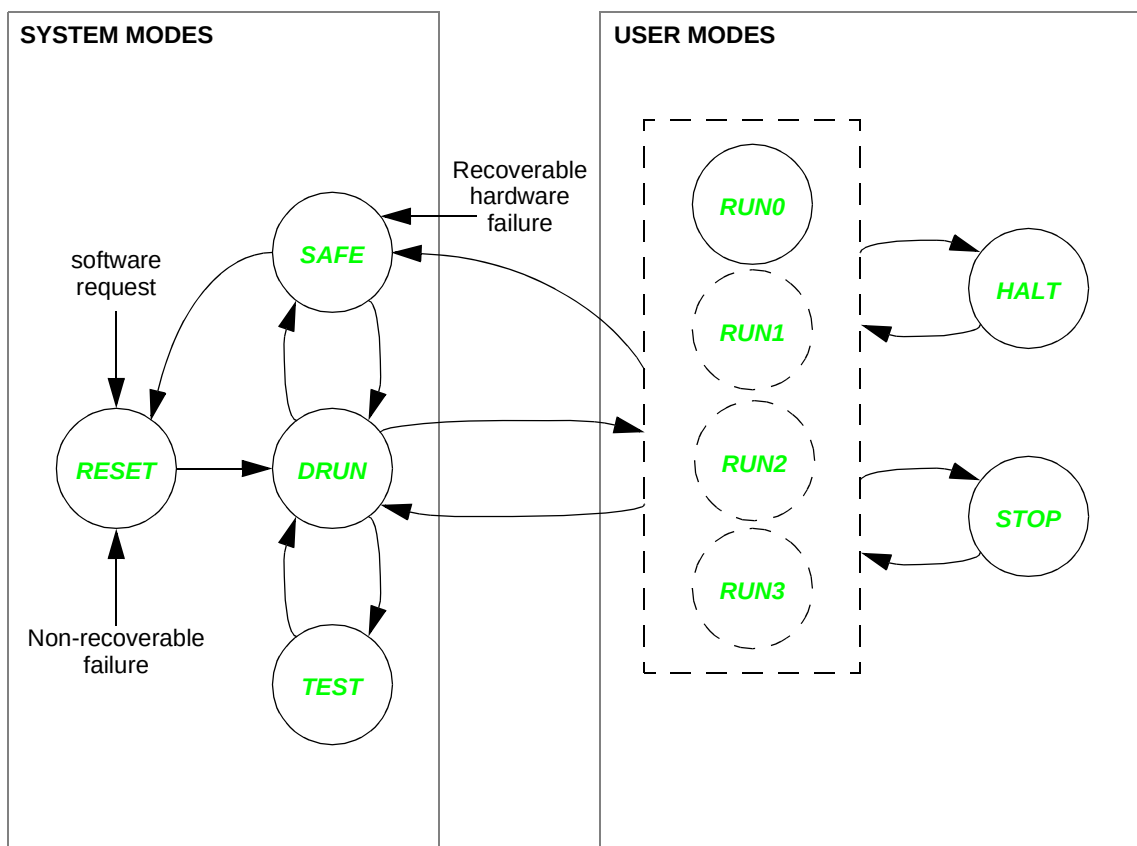
USER modes:

- USER modes control performance and consumption during normal application operation. The modes are organized according to the performance provided.
- The USER modes are RUNx, HALT, and STOP.
- Each mode is entered by writing a dedicated register protected by a key, to avoid unwanted transition.
- [Figure 50](#) shows the mode relationships.

Table 116. USER modes

Mode	Description	Entry	Exit
RUNx	These are software running modes where most of the processing activity is done. These run modes allow the user to enable different clock and power configurations of the system with respect to each other.	Software request from DRUN, interrupt event from HALT, wakeup request from STOP	Reset asserted, SAFE through S/W or H/W fail, RUNx, HALT, STOP, STANDBY through S/W
HALT	Reduced-activity low-power mode during which the core clock is disabled. It can be configured to switch off analog peripherals like PLL, flash, main voltage regulator, etc., for efficient power management at the cost of higher wakeup latency.	Software request from RUNx	Reset asserted, SAFE on hardware failure, RUNx on interrupt event
STOP	Advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including the oscillator for efficient power management at the cost of higher wakeup latency.	Software request from RUNx	Reset asserted, SAFE on hardware failure, RUNx on wakeup event

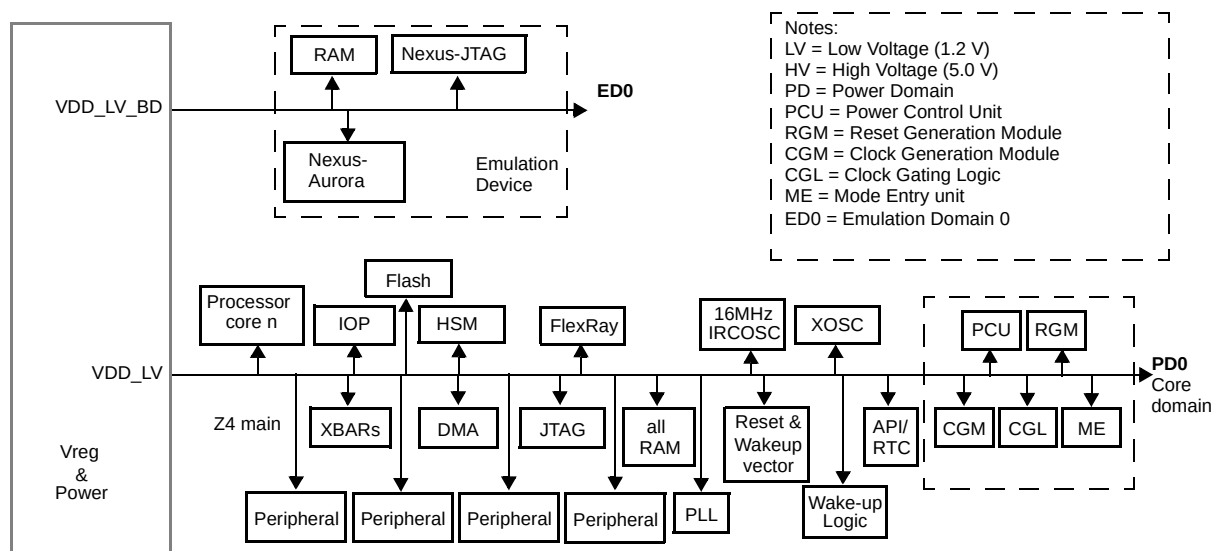
Figure 50. Mode relationships



10.1.1 Power management framework

The power management framework supports a variety of configuration options. See [Figure 51](#) for more details. SPC57EM80xx implements a unique core voltage power domain. There are no separate functional power domains.

Figure 51. Device power management framework



10.1.2 Power management supply description

Table 117 provides an overview of the PMC supply signals.

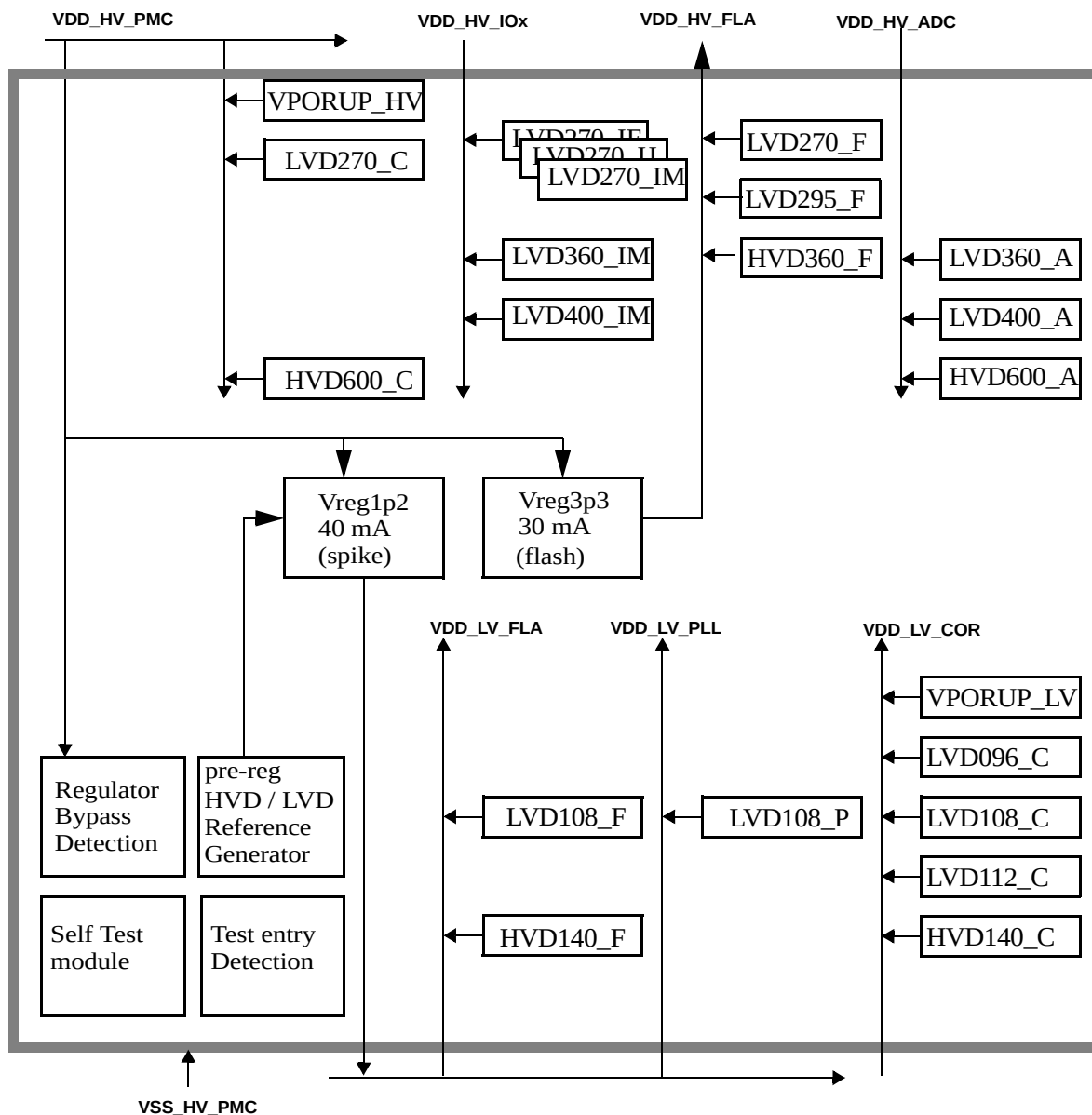
Table 117. Device power management controller external signals

Name	Type	Description
VDD_HV_ADR_D	Reference	Voltage reference of ADC Σ/Δ module
VDD_HV_ADR_S	Reference	Voltage reference of ADC SAR module
VDD_HV_ADV	Supply	High voltage supply for the ADC modules
VDD_HV_FLA	Decoupling	Decoupling supply pin for Flash
VDD_HV_IO	Supply	High voltage power supply for the I/Os
VDD_HV_IO_FLEX	Supply	FlexRay/Ethernet 3.3 V I/O supply
VDD_HV_IO_JTAG	Supply	Oscillator and JTAG pin supply
VDD_HV_PMC	Supply	High voltage power supply for internal power management unit
VDD_LV	Supply	Low voltage power supply for the core area
VDD_LV_BD	Supply	Low voltage power supply for the buddy device
VSS	Ground	Ground supply for the device / I/O. This is covering both VSS_LV and VSS_HV in case of exposed pad device.
VSS_HV_ADR_D	Reference	Ground reference of ADC Σ/Δ module
VSS_HV_ADR_S	Reference	Ground reference of ADC SAR module
VSS_HV_ADV	Supply	Ground supply for the ADC modules

Warning: There are constraints on some input voltages relative to other input voltages. Please refer to the device data sheet for detailed information.

10.1.3 SPC57EM80xx power management controller overview

Figure 52. SPC57EM80xx PMU overview



VDD_HV_IO is redundant, ensuring that single pin failure does not prevent device from functioning correctly. Full specification may not be guaranteed.

VDD_LV is redundant, ensuring that single pin failure does not prevent the device from functioning correctly.

10.1.3.1 1.2 V safety/spike regulator specification

SPC57EM80xx implements a 1.2 V safety/spike regulator. This regulator can be used to handle fast application current variation that cannot be fully handled by external regulator.

10.2 Low power mode support

The SPC57EM80xx microcontroller supports low power modes.

Dedicated logic prevents uncontrolled activation of low-power mode. This is implemented using protected sequence from ME.

Any register leading to low power entry is triple-voted.

10.2.1 Low power support and STOP mode implementation

Low power mode may be used to control the engine during the power-up phase, as in a Start-Stop system or when the key has been turned off.

The device implements the equivalent of an idle or power-down mode through STOP and HALT mode configuration.

The microcontroller application code can be used to put the device or modules into STOP mode.

A combination of mode control and clock control allows the user to configure the STOP mode.

In Afterrun mode, peripheral modules can be enabled/disabled in STOP using the control described in the ME section. For alternate configurations, other modules can be enabled/disabled in STOP using the control described in the ME section. External interrupt request channels and external wake-up channels can be made active to wake-up the device in alternative to peripherals interrupts.

All pins associated to the active modules are functional.

Control of the pins is defined by user during run time. A wake-up line is associated to each selectable module.

One of the available modules can be configured to perform the wake-up task. For example, with four CAN modules on the chip, three modules are set completely to power down mode and only one is configured for wake-up message receive (any of the four modules can be configured to perform this task).

All CAN modules can be independently configured to receive messages and respond to wake-up messages. Control of the pins is defined by user during run time.

In idle mode these modules can also be idle, but able to wake-up as soon as activity is detected and wake the core.

All these modules can be configured to a low power state. Wake-up recovery is programmable, for instance depending on state or activity of associated pin.

All parts of the device can be disabled. Following modules are configurable:

- There is one CAN module, selectable from any of the CAN modules on the device. The others can be disabled.
 - The maximum CAN baud rate is 1 MBaud.
 - One CAN module is able to receive messages from the CAN bus and wake-up the controller if the received message contains the correct message identifier.
- One LIN/UART module is able to recognize special wake-up messages on the receive line and then wake up the controller.
- Some “request” pins are able to initiate a wake-up. Refer to the pinout chapter for more information.
- The system timer is able to initiate a wake-up.

Configurable modules (ASC, three CANs, timer) can all be off in STOP mode.

The CAN module recovers from wake-up instantaneously but its ability to capture the message depends on settings of certain peripherals such as the CAN protocol clock. Depending on recovery time this impacts whether the first message can be captured, hence software must be used as a workaround.

Device implements all three mechanisms to exit from stop mode: pin wakeup, CAN wake-up and timed wake-up.

10.2.1.1 After-run implementation

After-run mode is implemented mainly making use of clock gating, module disabling, reduced frequency, mode switching.

Assuming that the selected CAN is kept enabled (clocked and active) then it can maintain its synchronization with the CAN network. Assuming also that the supplied CAN module can buffer the received (first) message, the CAN is then able to interrupt the IOP. The IOP would execute from TCM (i.e., Flash is disabled) and use the internal RC oscillator (IRCOSC) or the external oscillator (XOSC). To support this would keep the PMC, LVD, temperature sensor, internal FIRC, XOSC and pads active. The PLL, Flash, and ADC would be disabled, with rest of part static and clock gated off.

10.2.1.2 Checker core disabling

The checker core is always in lock step with the main core — hence if the main core is clocked, the checker core is clocked. Also if the main core is gated off, then the checker core is also gated off.

It is possible to disable the safety checker core to reduce current consumption.

The checker core can be placed in idle mode throughout the afterrun mode. By default the checker core mimics the main core, but it can be clock gated off during afterrun mode whilst continuing to clock the main core.

The checker core is implemented in such a way that its entire clock tree can be gated off while the core logic remains supplied.

The checker core (part of the Checker Logic) and the corresponding ‘checked main’ core share the same cache SRAM arrays and local SRAM arrays. Once the checker logic is disabled (i.e., its clock is gated) the SRAM array of the cache and the local memory of the main core are still usable by the main core.

10.3 Flash power requirements

During start-up operation, the flash module operates from an untrimmed regulator, which has a slightly larger variation than the ideal case. During this start-up phase, the flash can still be read, but it must be read at a slower rate. For SPC57EM80xx, read is performed differentially, i.e., the data is duplicated / inverted across two locations. While operating in this slower read mode, the flash can operate at a much reduced voltage, and hence the untrimmed regulator should default to a < 3.3V average. This will help guarantee that the maximum value is not exceeded.

Following completion of the start-up phase the regulator is in its trimmed condition, and subsequently the flash module reference current has also been trimmed. Therefore when entering normal operation the flash can be read in the normal access manner and support the full flash specification.

For SPC57EM80xx, the flash is using the externally supplied 1.25 V for the low voltage supply, while high voltage (3.3 V) is generated by a dedicated internal regulator.

10.4 Device trimming

During the initialization phase the device defaults to a pre-determined state for each of the LVDs, HVDs and the internal regulators. As the flash becomes available the differential read process allows the trimmed data to be available for trimming the internal LVDs, HVDs and regulators.

Please refer to [Section 10.6, Power sequence](#) for further details.

10.5 Supply monitoring (POR and LVDs)

The function of the POR and LVD circuits is to hold the device in reset regardless of how slow the supply voltage rise is, until the point at which the POR and LVDs are released.

The POR and LVD circuits function correctly even if the input voltage is non monotonic.

10.5.1 Power-on reset (POR)

The PMC implements two internal power-on reset circuits:

- VPORUPLV monitors the voltage on the 1.2 V input supply. It is monitoring the VDD_LV_CORE pin. The VPORUPLV asserts a reset when the input supply is below defined values.
- VPORUPHV monitors the voltage on the 5.0 V input supply. It is monitoring the VDD_HV_PMC pin. The POR trip point is high enough to make sure all the LVD circuits are functional.

See [Chapter 8: Reset and Boot](#) for PORST/ESR0 pin functionality.

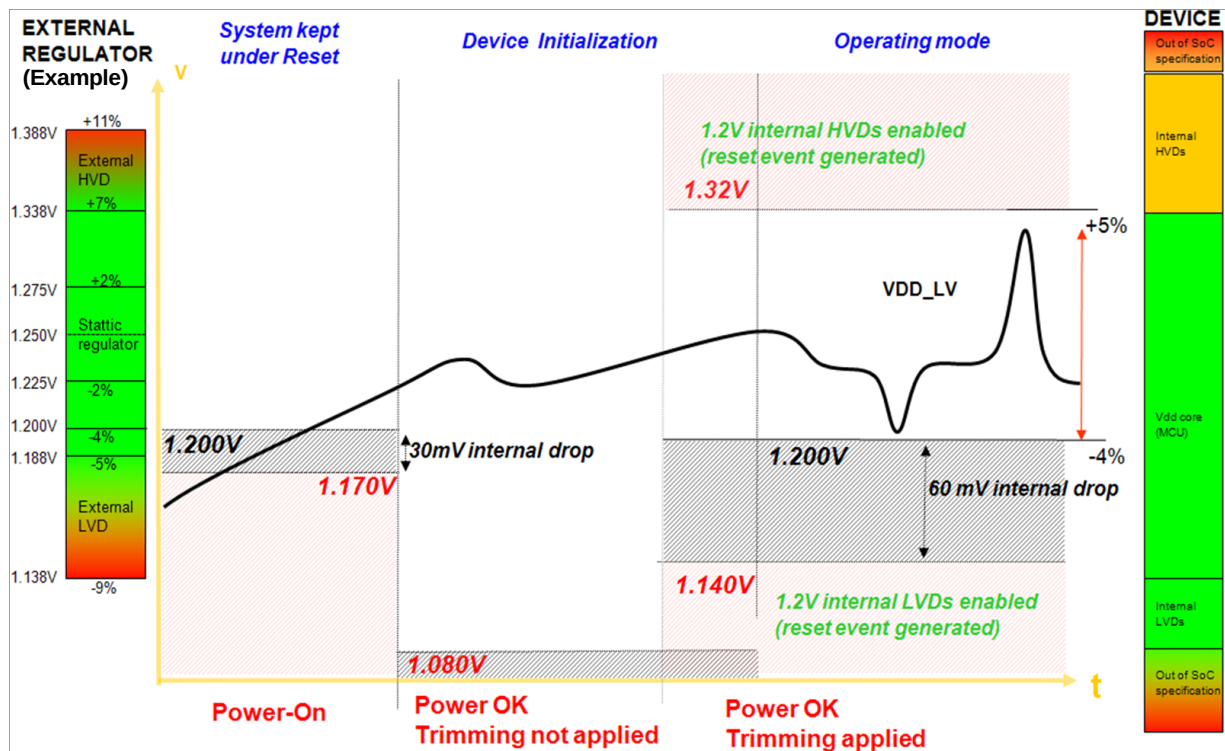
10.5.2 Behavior of device LVD / HVD

Although there is an option to disable the LVD and HVD following reset (see section [Section 10.5.3, Low Voltage Detection \(LVD\) and High Voltage Detection \(HVD\)](#)), they are capable of being used in a 'monitor' only mode and also capable of generating a safe / interrupt event. The diagram in [Figure 53](#) illustrates the behavior of these internal LVD /

HVD circuits during the power on phase, through into device initialization and subsequently in 'normal' run / operating mode.

The LVDs/HVDs can also be configured after device initialization preventing reset to happen when supply crosses the LVD threshold, effectively providing a higher voltage operating range. It is the responsibility of the application to ensure that the device remains in the functional range.

Figure 53. Internal LVD and HVD in configuration example during power-up



10.5.3 Low Voltage Detection (LVD) and High Voltage Detection (HVD)

The internal LVD circuits monitor when the voltage on the corresponding supply is below defined values and either assert a reset or an interrupt. The LVDs also support hysteresis in the falling and rising trip points.

10.5.3.1 LVD and HVD implementation

- All LVDs and HVDs are capable of generating either an RGM and/or FCCU event, with the following exceptions:
 - The LVD270 (that is, the LVD monitoring the high voltage supply input) can be disabled only for test purpose and should otherwise always generate a reset when triggered.

Warning: In case LVD270 are disabled, the device may not detect through internal monitor that VDD_HV voltage is outside

functional range. If this event occurs, reset state must be ensured by assertion of external PORST pin from the user.

- It is not possible to disable the LVD096 (that is, the LVD monitoring the internal core voltage) and they always generate a reset when triggered. This LVD096 is integral to the POR management.
- All LVDs and HVD configured for reset generation can cause functional or destructive reset. MC_RGM PHASE0 is not exited until all destructive reset conditions are cleared.
- The appropriate bits in the PMC registers are set by LVD and HVD events.
- The LVDs and HVDs control is protected by the SoC wide register protection scheme. Therefore it is configurable as long as the scheme is followed. Please refer to [Chapter 83: Register Protection \(REG_PROT\)](#).
- SPC57EM80xx implements REE DCF records to allow to associate “configurable” LVDs/HVDs to a RESET event. This is a write once mechanism managed by SSCM during device initialization. When RESET event is selected through REE DCF record, it cannot be changed until next RESET event. When RESET event is not selected through REE DCF record, LVDs/HVDs trigger event default to FCCU event.

Note: The list of the “configurable” LVDs/HVDs is provided in [Table 118](#), in which the “User configurability” columns provide the possible user configuration.

- When the LVD or the HVD is enabled for destructive reset generation and a subsequent trigger event is detected, the external PORST pin is driven low.

Note: LVD096 and LVD270 (low range LVDs) cannot be disabled by the user during boot sequence. These modules are used during power-up phase and must ensure that an absolute lowest threshold of operation is never crossed. This is not a guarantee that the device will function down to this level. It is rather a guarantee that the device will recover if this level is crossed.

Warning: The LVD270 can be disabled for test purpose through software accessing PMC_REE register. In case LVD270 are disabled, the device may not detect through internal monitor that VDD_HV voltage is outside functional range (below LVD270 threshold). If this event occurs, reset state must be ensure by assertion of external PORST pin from the user. When supply goes below the threshold of the POR240 (internal power-on state detector), device will be forced into reset and LVD270 is automatically re-enabled.

Table 118. POR and voltage monitors description

Monitor name	Register bit	Description	Monitor/ sensing	POWERUP- PHASE0	PHASE1-2-3	User configurability	
						DCF (PHASE3)	REE/FEE / IE ⁽¹⁾ (RUN)
POWER MANAGEMENT UNIT VOLTAGE MONITORS – LOW VOLTAGE SUPPLY							
LVD096_C	VD2	Low voltage supply core low range low voltage detector	VDD_LV_CORE (hot-point)	ENABLE	ENABLE	ENABLE	ENABLE
LVD096_F	VD2	Low voltage supply flash low range low voltage detector	VDD_LV_FL A (hot-point)	ENABLE	ENABLE	ENABLE	ENABLE
LVD108_C	VD3	Low voltage supply core medium range low voltage detector	VDD_LV_CORE (hot-point)	DISABL E	DISABLE	LVDM[0]	VD3_C
LVD108_F	VD3	Low voltage supply flash medium range low voltage detector	VDD_LV_FL A (hot-point)	DISABL E	DISABLE	LVDM[1]	VD3_F
LVD108_P	VD3	Low voltage supply PLL medium range low voltage detector	VDD_LV_PLL (hot-point)	DISABL E	DISABLE	LVDM[10]	VD3_P
LVD112_C	VD4	Low voltage supply core medium range low voltage detector	VDD_LV_CORE (cold-point)	ENABLE	DISABLE	LVDM[2]	VD4_C
HVD140_C	VD7	Low voltage supply core high voltage detector	VDD_LV_CORE (cold-point)	DISABL E	DISABLE	LVDM[3]	VD7_C
HVD140_F	VD7	Low voltage supply flash high voltage detector	VDD_LV_FL A (hot-point)	DISABL E	DISABLE	LVDM[12]	VD7_F
HVD145_C	VD8	Low voltage supply core high voltage detector	VDD_LV_CORE (cold-point)	ENABLE	ENABLE	LVDM[13]	VD8_C
POWER MANAGEMENT UNIT VOLTAGE MONITORS – HIGH VOLTAGE SUPPLY							
LVD270_C	VD9	High voltage supply core low range low voltage detector	VDD_HV_PMC (hot-point)	ENABLE	ENABLE	ENABLE	VD9_C
LVD270_F	VD9	High voltage supply flash low range low voltage detector	VDD_HV_FL A (hot-point)	ENABLE	ENABLE	ENABLE	VD9_F
LVD270_IM	VD9	High voltage supply I/O main low range low voltage detector	VDD_HV_IO_MAIN (hot-point)	ENABLE	ENABLE	ENABLE	VD9_IM

Table 118. POR and voltage monitors description (continued)

Monitor name	Register bit	Description	Monitor/ sensing	POWERUP-PHASE0	PHASE1-2-3	User configurability	
						DCF (PHASE3)	REE/FEE / IE ⁽¹⁾ (RUN)
LVD270_IF	VD9	High voltage supply I/O FlexRay low range low voltage detector	VDD_HV_IO_FLEX (hot-point)	DISABLE	DISABLE	ENABLE	VD9_IF
LVD270_IJ	VD9	High voltage supply I/O JTAG low range low voltage detector	VDD_HV_IO_JTAG (hot-point)	ENABLE	ENABLE	ENABLE	VD9_IJ
LVD270_O	VD9	High voltage supply I/O oscillator low range low voltage detector	VDD_HV_IO_JTAG (hot-point)	ENABLE	ENABLE	ENABLE	VD9_O
LVD295_F	VD10	High voltage supply flash medium range low voltage detector	VDD_HV_FLA (hot-point)	ENABLE	ENABLE	LVDM[4]	VD10_F
LVD295_A	VD10	High voltage supply flash medium range low voltage detector	VDD_HV_ADV (hot-point)	ENABLE	ENABLE	LVDM[14]	VD10_A
HVD360_F	VD12	High voltage supply flash low range low voltage detector	VDD_HV_FLA (hot-point)	ENABLE (security)	ENABLE (security)	ENABLE	VD12_F
LVD360_IM	VD13	High voltage supply I/O main medium range low voltage detector	VDD_HV_IO_MAIN (hot-point)	ENABLE	DISABLE	LVDM[5]	VD13_IM
LVD400_IM	VD14	High voltage supply I/O main high range low voltage detector	VDD_HV_IO_MAIN (hot-point)	ENABLE	DISABLE	LVDM[7]	VD14_IM
LVD400_A	VD14	High voltage supply ADC high range low voltage detector	VDD_HV_ADV (hot-point)	DISABLE	DISABLE	LVDM[8]	VD14_A
HVD600_C	VD15	High voltage supply PMC main high voltage detector	VDD_HV_PMC (cold-point)	DISABLE	DISABLE	LVDM[9]	VD15_C
HVD600_A	VD15	High voltage supply ADC main high voltage detector	VDD_HV_ADV (cold-point)	DISABLE	DISABLE	LVDM[11]	VD15_A

1. REE: Reset event enable / FEE: FCCU event enable / IE: Interrupt enable — See [Chapter 60: Power Management Controller digital interface \(PMC_dig\)](#).

[Table 119](#) provides the monitor status depending on configuration.

Table 119. Voltage monitors configurability

POWERUP	PHASE0-1-2	DCF	REE ⁽¹⁾	RES ⁽²⁾	FEE ⁽³⁾	IE ⁽⁴⁾	Monitor status
ENABLE	ENABLE	—	—	—	—	—	Default configuration (boot threshold monitor) Monitor enabled during full run from power-up to power-down
ENABLE	DISABLE	ENABLE	ENABLE	DEST	—	—	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 entry. Monitor re-enabled during phase 3 on DCF record reading. Trigger destructive event reset during reset flow. Configuration is locked to enable during run time.
ENABLE	DISABLE	ENABLE	ENABLE	FUNC	ENABLE	—	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 entry. Monitor re-enabled during phase 3 on DCF record reading. Trigger destructive event reset during reset flow. Configuration is locked to enable during run time. After reset completion: LVD event demote from destructive reset to functional reset by software after completion of reset sequence. FCCU event are generated on functional reset.
ENABLE	DISABLE	ENABLE	ENABLE	FUNC	DISABLE	—	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 entry. Monitor re-enabled during phase3 on DCF record reading. Trigger destructive event reset event. Configuration is locked to enable during run time. After reset completion: LVD event demote from destructive reset to functional reset by software after completion of reset sequence. FCCU event are not generated.

Table 119. Voltage monitors configurability (continued)

POWERUP	PHASE0-1-2	DCF	REE ⁽¹⁾	RES ⁽²⁾	FEE ⁽³⁾	IE ⁽⁴⁾	Monitor status
ENABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	ENABLE	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 entry. Monitor re-enabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. After reset completion: ⁽⁵⁾ The LVD event is also associated to an interrupt event.
ENABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	DISABLE	Default configuration (hysteresis monitor). Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 entry. Monitor re-enabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. After reset completion: ⁽⁵⁾ Configuration can be changed by software to any of following events: functional reset, destructive reset, FCCU or interrupt event by software.
ENABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	ENABLE	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 entry. Monitor re-enabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. After reset completion: ⁽⁵⁾ The LVD event is associated to an interrupt event whereas FCCU associated event is disabled.

Table 119. Voltage monitors configurability (continued)

POWERUP	PHASE0-1-2	DCF	REE ⁽¹⁾	RES ⁽²⁾	FEE ⁽³⁾	IE ⁽⁴⁾	Monitor status
ENABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	DISABLE	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 entry. Monitor re-enabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. After reset completion: ⁽⁵⁾ The FCCU event associated to LVD event is disabled. After the software configuration, an LVD event will generate no event.
DISABLE	DISABLE	ENABLE	ENABLE	DEST	—	—	Monitor enabled during phase 3 on DCF record reading. Trigger reset event. Configuration locked to enable during run time.
DISABLE	DISABLE	ENABLE	ENABLE	FUNC	ENABLE	—	Monitor enabled during phase 3 on DCF record reading. Trigger reset event. Configuration locked to enable during run time. After reset completion: LVD event demote from destructive reset to functional reset by software after completion of reset sequence. FCCU event are generated on functional reset.
DISABLE	DISABLE	ENABLE	ENABLE	FUNC	DISABLE	—	Monitor enabled during phase 3 on DCF record reading. Trigger reset event. Configuration locked to enable during run time. After reset completion: Reset event demote from destructive reset to functional reset by software after completion of reset sequence. FCCU event are not generated.
ENABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	ENABLE	Monitor enabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. After reset completion: ⁽⁵⁾ The LVD event is also associated to an interrupt event.

Table 119. Voltage monitors configurability (continued)

POWERUP	PHASE0-1-2	DCF	REE ⁽¹⁾	RES ⁽²⁾	FEE ⁽³⁾	IE ⁽⁴⁾	Monitor status
DISABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	DISABLE	Default configuration (application monitor). Monitor enabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. After reset completion: Configuration can be changed by software to any of following events: functional reset, destructive reset, FCCU or interrupt event by software.
DISABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	ENABLE	Monitor enabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. After reset completion: ⁽⁵⁾ The LVD event is associated to an interrupt event whereas FCCU associated event is disabled.
DISABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	DISABLE	Monitor enabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. After reset completion: ⁽⁵⁾ The FCCU event associated to LVD event is disabled. After the software configuration, an LVD event will generate no event.

1. REE: Reset event enable

2. RES: Reset event select

3. FEE: FCCU event enable

4. IE: Interrupt enable

5. Configuration can be changed by software to any of following events: functional reset, destructive reset, FCCU or interrupt event by software

10.6 Power sequence

The following sections describe the power sequence and the relation between the different supplies during power-up and power-down.

The device is considered to be in a power sequence (or POWERUP state) when it is either not supplied or partially supplied. An internal power-on signal is used to identify POWERUP

state. This signal is released high on exit of power sequence. The power-on signal is a combination of the LVDs monitoring the following supplies:

- VDD_LV
- VDD_HV_PMC
- VDD_HV_IO
- VDD_HV_FLA

The actual threshold use for each LVD is dependent on the configuration of the device. This is configurable by hardware (flash option bits content) or by software (LVD event configuration through register interface). Once power-on signal has been asserted, device configuration is reset to default power-up configuration.

10.6.1 Power-up sequence

In this section it is assumed that all supplies are low when entering the power-up sequence. Brown-out and power down sequences are managed in specific sections.

10.6.1.1 POWERUP state

At the beginning of power-up, the internal power-on signal remains low due to internal parasitics diodes. As soon as minimum threshold is reached on VDD_LV and VDD_HV supply, the internal power-on signal is forced low until power-up LVDs reach upper not trimmed threshold (refer to datasheet).

The different supplies can rise independently as long as the constraints described in the datasheet are met.

During power-up, all functional terminals are maintained into a known state as described in [Table 120](#).

Table 120. Functional terminal state during power-up and reset

Terminal type ⁽¹⁾	POWERUP ⁽²⁾ pad state	RESET pad state	DEFAULT pad state ⁽³⁾	Comments
PORST	strong pull-down	weak pull-down	weak pull-down	Power-on reset pad
ESR0	strong pull-down	strong pull-down	weak pull-up ⁽⁴⁾	Functional reset pad
ESR1	weak pull-up	weak pull-up	weak pull-up	Functional NMI/reset pad
TEST_MODE	weak pull-down	weak pull-down	weak pull-down	—
GPIO	weak pull-up	weak pull-up	weak pull-up	—
ANALOG	high impedance	high impedance	high impedance	—
ERROR	high impedance	high impedance	high impedance	During functional reset, pad state can be overridden by FCCU
JCOMP ⁽⁵⁾	high impedance	weak pull-down	weak pull-down	—
TCK ⁽⁵⁾	high impedance	weak pull-down	weak pull-down	—
TMS ⁽⁵⁾	weak pull-up	weak pull-up	weak pull-up	—

Table 120. Functional terminal state during power-up and reset (continued)

Terminal type ⁽¹⁾	POWERUP ⁽²⁾ pad state	RESET pad state	DEFAULT pad state ⁽³⁾	Comments
TDI ⁽⁵⁾	weak pull-up	weak pull-up	weak pull-up	—
TDO	high impedance	high impedance	high impedance	—

1. Refer to pinout information for terminal type
2. POWERUP state is maintained until supply cross the power-on reset threshold: V_{PORUP_LV} for LV supply, V_{PORUP_HV} for high voltage supply.
3. Before software configuration
4. ESR0 configuration move from strong pull-down to weak pull-up on completion of internal reset sequence (PHASE3[FUNC]).
5. JTAG interface implements build-in pull-up/pull-down configuration to ensure noise immunity. Worst case is boundary scan activation, requiring JCOMP to pulse high and following events to occur during JCOMP pulse high:
 - 23 TCK events corresponding to 12 clock cycles.
 - 3 TMS events synchronous with TCK: sequence 0,1,1,0,0,0,0,0,0,0,1
 - 2 TDI events synchronous with TCK: sequence x,x,x,x,x,0,0,0,1,0,0,x
 TCK, TMS, TDI are implementing hysteresis requiring minimum variation

Digital reset is asserted, ensuring that all modules registers are reset to power-on value.

10.6.1.2 POWERUP exit

The POWERUP state can be exited when both VDD_LV and VDD_HV are above the internal LVD thresholds.

10.6.1.2.1 VDD_LV conditions and LVD trimming/enabling sequence

During POWERUP, only the following low voltage LVDs are enabled:

- LVD112_C
- LVD096_C, LVD096_F

The VDD_LV conditions to exit POWERUP are the following:

- LVD096_C, LVD096_F upper threshold is crossed
- LVD112_C upper threshold is crossed

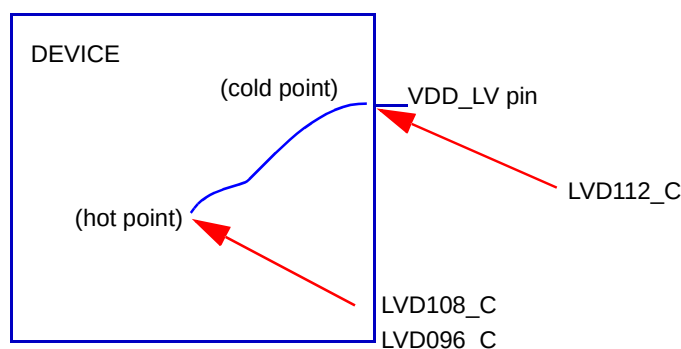


Figure 54. VDD_LV monitored voltages during power-up

After the POWERUP exit conditions (low voltage, high voltage conditions) have been verified, the internal power-on signal is released to all analog modules.

The internal RC oscillator module (IRCOSC) starts initialization and provides clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles and LVD112_C is masked. Only LVD096_C remains active.

LVD096_C and LVD112_C share the same reference. $LVD112_C - LVD096_C > 90\text{ mV} \pm 5\%$. This difference provides margin with respect to maximum internal resistive drop (40 mV at I_{max}) and 50 mV hysteresis addressing external supply drop.

The device proceeds through the reset sequence through RGM phase PHASE0, PHASE1[DEST], PHASE2[DEST] and PHASE3[DEST].

Voltage detector (LVD/HVD) modules are trimmed at the beginning of PHASE3[DEST]. Trimming of LVDs/HVDs are stored as internal DCF records. They are read by SSCM at low voltage, using the differential flash read accesses and applied to each LVDs/HVDs module. After analog delay ($t_{LVDTRIM}$) has elapsed, the PMC acknowledges that all LVDs/HVDs have been trimmed and supply is above threshold, the SSCM proceeds with the reset sequence, eventually running full speed accesses to the flash to read the option bits required to complete device configuration.

The configurable LVD/HVD modules can optionally be unmasked at the end of PHASE3[DEST]. Mask information is read as DCF record from the flash UTEST option bits.

When LVD108_C and LVD360_IM are masked by the application using the flash user option bits, the device should rely on PORST signal to detect a voltage failure during power-up. The device must wait for PORST to be released high before proceeding with the power-up sequence. This may increase the amount of time necessary to complete the reset sequence.

[Figure 55](#) provides an example of VDD_LV power-up sequence from POWERUP to PHASE3[DEST].

[Figure 56](#) illustrates threshold variation of VDD_LV LVD monitor during power-up sequence from POWERUP to PHASE3[DEST]. Internal LVD108_C LVD and internal HVD140_C HVD are *enabled* during PHASE3[DEST].

[Figure 57](#) illustrates threshold variation of VDD_LV LVD monitor during power-up sequence from POWERUP to PHASE3[DEST]. Internal LVD108_C LVD and internal HVD140_C HVD are *disabled* during PHASE3[DEST]. The device relies on external voltage monitors.

Figure 55. Example of VDD_LV power-up sequencing

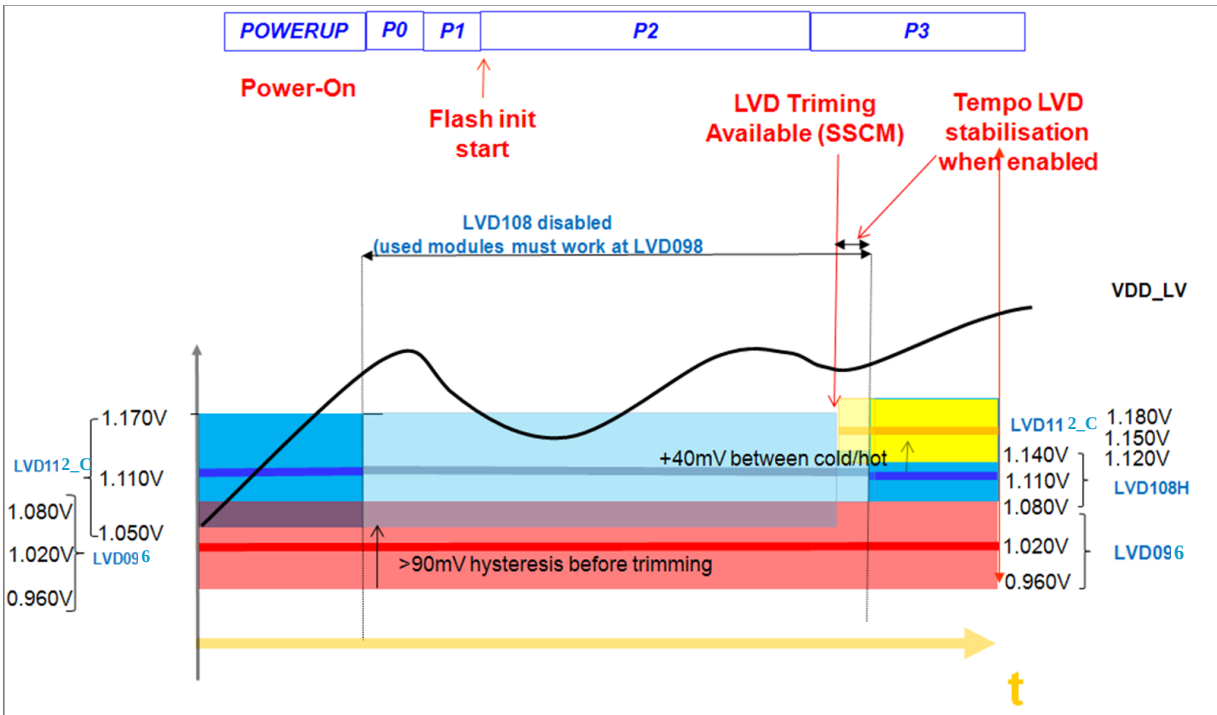


Figure 56. Threshold variation during power-up sequence: hot point LVD enabled

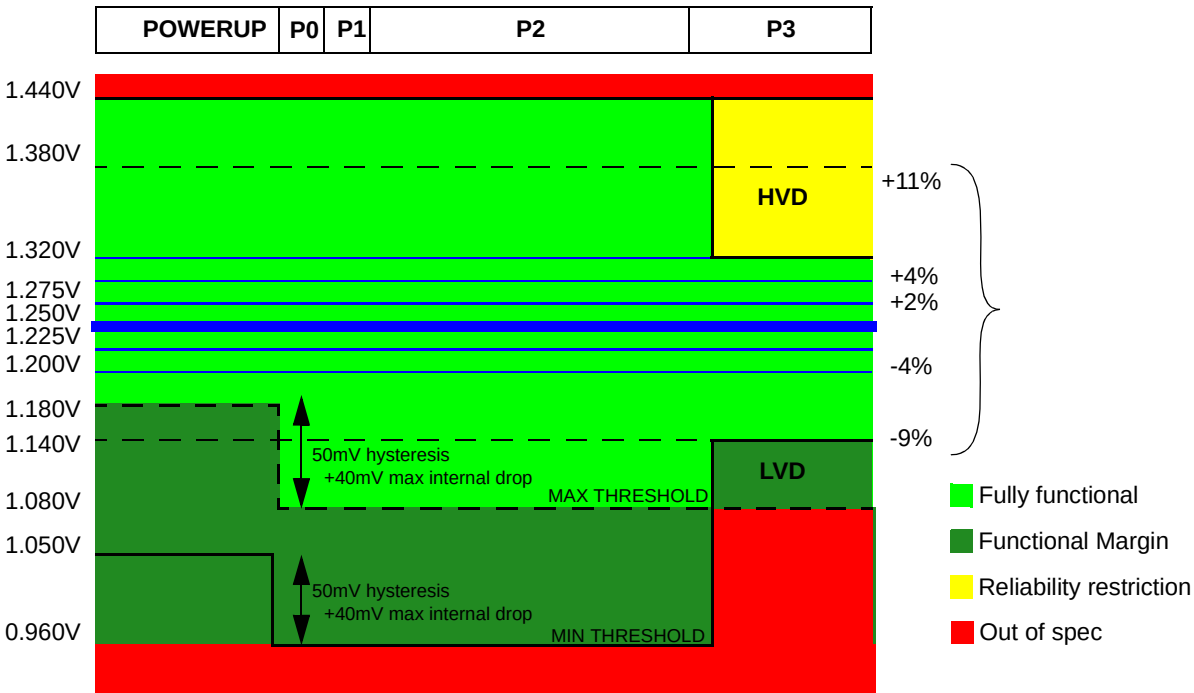
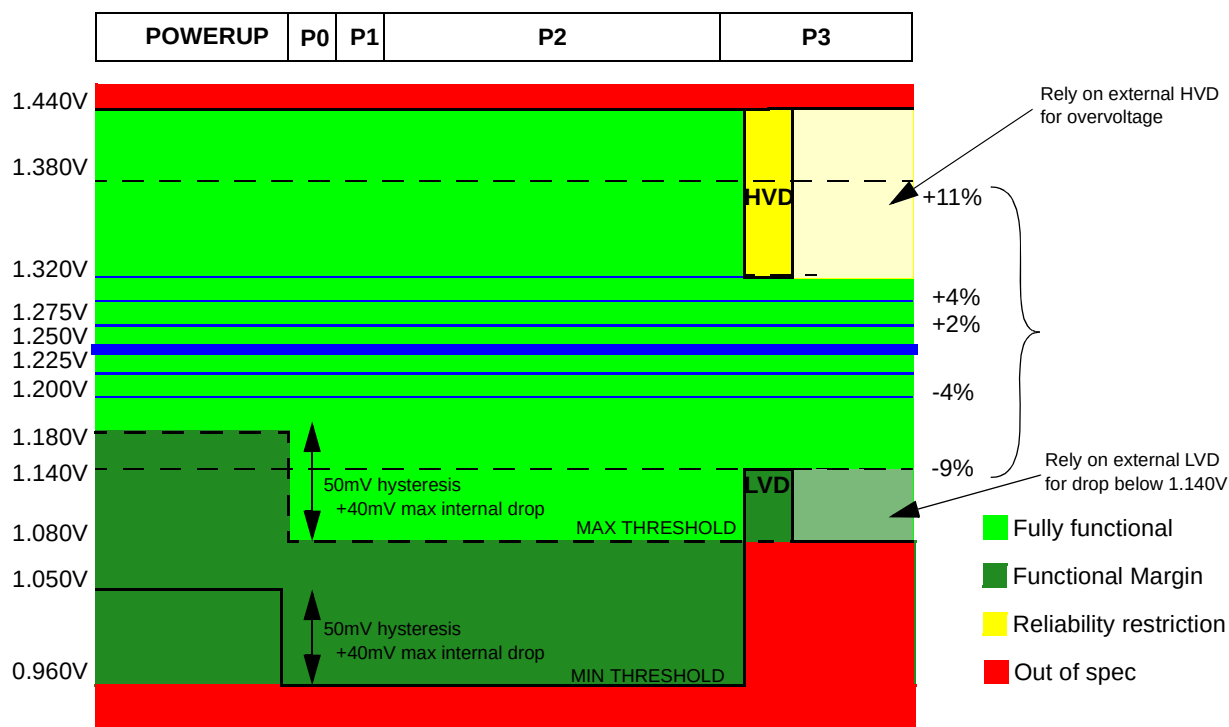


Figure 57. Threshold variation during power-up sequence: hot point LVD disabled



10.6.1.2.2 VDD_HV conditions and LVD trimming/enabling sequence

During POWERUP, the following high voltage LVDs are enabled:

- LVD400_IM
- LVD360_IM
- LVD295_F
- LVD270_xx

The VDD_HV conditions to exit POWERUP are the following:

- LVD400_IM upper threshold is crossed.
- LVD360_IM upper threshold is crossed.
- LVD300_F upper threshold is crossed.
- LVD270_xx upper threshold is crossed.

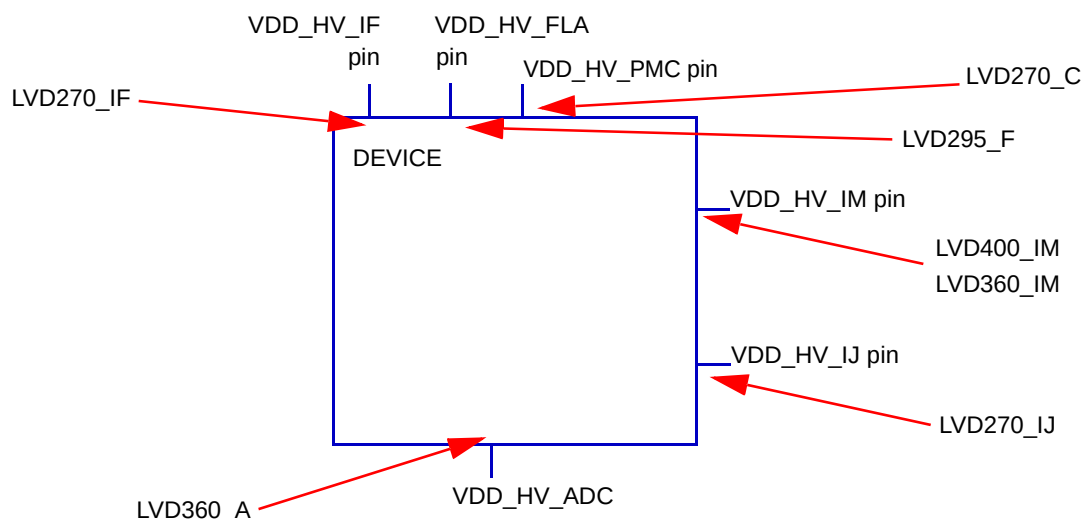


Figure 58. VDD_HV monitored voltages during power-up

After both LV and HV POWERUP exit conditions have been verified, ipp_por_en_b signal is released to all analog modules.

The IRCOSC module starts initialization and provides the clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles and LVD400_IM is masked. All other HV LVDs remain active.

LVD360_IM and LVD400_IM share the same reference. LVD400_IM - LVD360_IM difference is implementing margin with respect to maximum internal resistive drop and hysteresis addressing external supply drop.

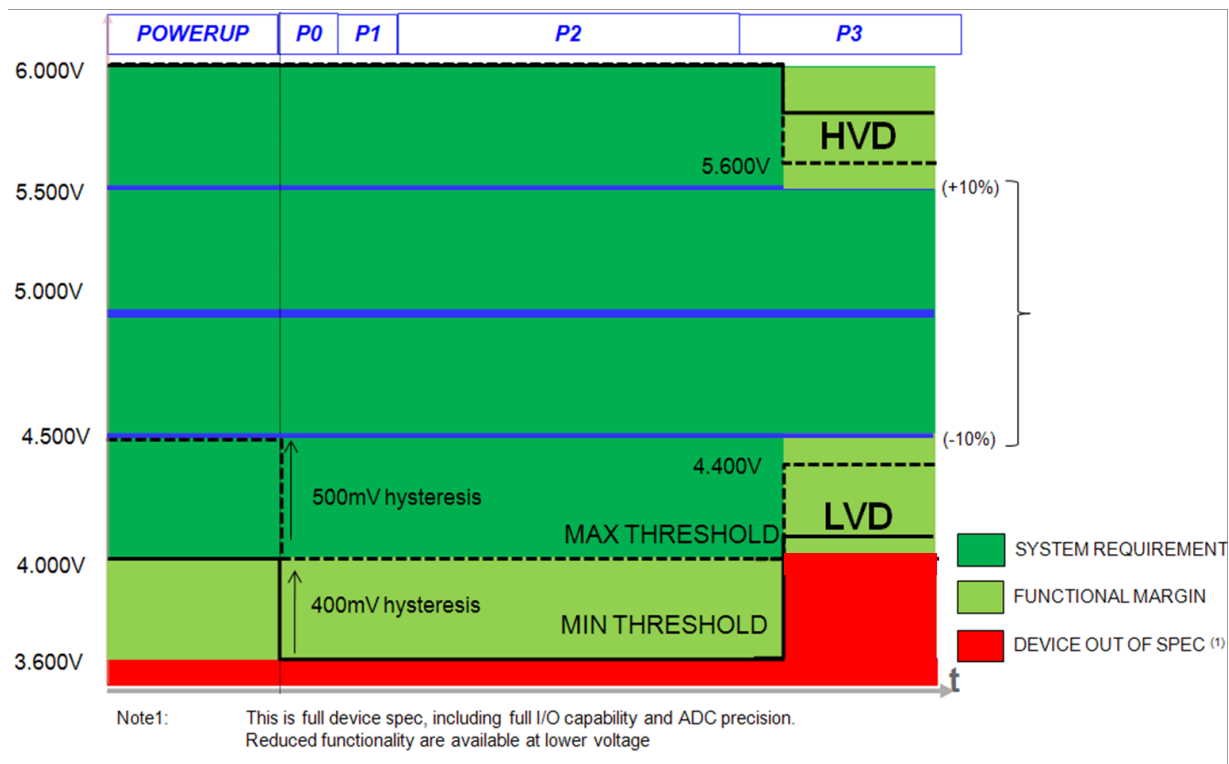
Device proceeds through the reset sequence through RGM phase PHASE0, PHASE1[DEST], PHASE2[DEST], PHASE3[DEST].

LVD/HVD modules are trimmed at the beginning of PHASE3[DEST]. Trimming is done by SSCM at low voltage, using the differential flash read accesses. After LVD is completed, SSCM waits for PMC acknowledgement before proceeding with reset sequence.

Configurable LVD/HVD modules are optionally enabled at the end of PHASE3[DEST].

[Figure 59](#) provides the threshold variation of VDD_HV LVDs monitor during power-up sequence from POWERUP to PHASE3[DEST]. Internal LVD360_IM LVDs and internal HVD600_IM HVD are *enabled* during PHASE3[DEST].

Figure 59. Threshold variation during power-up sequence — 2.7V–5.5V hot point LVD disabled, 4.5V–5.5V range



The device needs to fulfill all conditions described on [Section 10.6.1, Power-up sequence](#) to recover.

10.6.2 Power-down sequence

In case the threshold of the higher ranges LVDs that are configured to generate destructive reset is crossed but neither LVD096 nor LVD270 threshold is crossed, the device enters PHASE0 phase.

Power-down sequence is actually entered as soon as the threshold of one of the LVD096 or LVD270 is crossed. The device enters the POWERUP state. The internal power-on signal `ipp_por_en_b` is asserted.

On `ipp_por_en_b` assertion, all analog modules reach their safe state.

Device supplies may then proceed to drop down to ground either through device leakage or external pull-down. During power-down, all supplies should comply with supply constraints.

10.6.3 Brown-out management

During brown-out, the device re-enters the POWERUP phase as soon as the threshold of either LVD096 or LVD270 is crossed. The internal power-on signal `ipp_por_en_b` is asserted.

Note: The device will correctly start independently from any residual voltage. Below LVD270 threshold the device is under POWERUP state. Above the LVD270 threshold, the device is

into a RESET state as defined by Voltage monitor or external PORST pin or active state when all have been released.

Warning: This mechanism is ensured when LVD270 are not disabled by the user. In case LVD270 are disabled, reset state must be ensured by assertion of external PORST pin from the user. When supply goes below the threshold of the POR240 (internal power-on state detector), device will be forced into reset and LVD270 is automatically re-enabled.

On internal power-on signal assertion, all analog modules reach their safe state.

10.6.4 Low voltage requirement during crank

The device can continue operation to the minimum input voltage during crank.

In order to proceed with execution during cranking and prevent device reset, it is important to correctly configure the high voltage LVDs.

Internal LVDs that monitor the supply ensure that the flash content is protected.

11 Security

All SPC57xx family devices have a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access. Some security features are available in all device configurations while other more advanced security features are not available in all device configurations. These more advanced features require more than simply being turned on or off; they require significant involvement for implementation. Consequently, they provide an important opportunity to define the security level of the device.

11.1 Basic security

All SPC57EM80xx devices have the following basic set of security features.

- Device censorship based on the life cycle model with code and data access progressively more restricted as device matures through defined life cycle steps
- Memory security features:
 - NVM censorship support, password protection, one-time-programmable (OTP) flash memory areas, Flash erase counter and tamper detection
 - SRAM and caches initialized to a constant value after reset (power-on reset, when in a test mode, when not booting from internal flash memory); MBIST functionality used in case of power-on and destructive resets to initialize RAMs selected via configuration in UTEST flash; BAM software routines to initialize RAMs when not booting from internal flash memory
- Monitoring of operating conditions
- Unique ID for each device: each SPC57EM80xx device has a unique identification number stored in an OTP flash memory area which can be read by any of the cores on the device
- Secure watchdog timer
- Basic debugger restrictions (on/off via censorship mode)

11.2 Advanced security

Depending on the device configuration, the following features are available:

- Customer programmable Hardware Security Module (HSM): a dedicated security subsystem that includes a processor core, dedicated SRAM, an encryption module, and exclusive access to secure areas of device flash memory; HSM runs code independently from the main device processor cores and can be used to implement advanced security and monitoring functions
- Secure debugger interface
- Boot modes:
 - Trusted/secure boot support
 - Handshake with Boot Assist Module (BAM) supported

11.3 Detailed security information

Details of most of the SPC57EM80xx security features are published in a separate *SPC57EM80xx Microcontroller Security Reference Manual*, which is only available to qualified customers.

12 Calibration and Debug

This chapter discusses the device-specific information for the debug and calibration modules. This device implements a high-speed serial Nexus trace auxiliary port. The Aurora interface allows the Nexus protocol information to be transmitted serially at high speed over four Aurora lanes. The Aurora protocol handles the encoding of the data and striping the information across the four Aurora lanes available on the device. In addition, the LFAST module, a high speed calibration interface, is supported for run control and memory access. The LFAST signals are alternate functions on the device's JTAG pins. The device starts up in JTAG mode and the tool can request that the JTAG interface be converted to the LFAST high-speed interface. All standards used have freely available specifications for tool developers.

The debug and calibration features for this device are provided in two separate dies: the Production die and the Buddy die, or Buddy Device (BD). The Production die is packaged alone in the Production Device (PD) package. The PD is also assembled into an Emulation and Debug Device (ED) package that consists of the Production die along with the Buddy die. The Buddy device supports additional debug capability. The Nexus Aurora interface is only available on the ED. The PD only supports trace to the internal trace/calibration memory.

The Event Out ($\overline{\text{EVTO}}[0,1]$) signals are connected to the external TGOUT[0,1] device pins and the Event In (EVTI) signals are connected to the external TGIN[0,1] device pins.

12.1 Core debug support

Internal debug support in the e200z420n3 and e200z425n3 cores allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. The cores have the capability of exiting reset halted in debug mode, such that no code runs. This allows debuggers to perform operations such as configuration of calibration remap or edit of overlay memory prior to user code execution.

For details, see [Chapter 63: e200z420n3 Core Debug Support](#) and [Chapter 64: e200z425n3 Core Debug Support](#).

12.2 Run control and memory access

Run control is any operation required to control device operation. Run control includes starting and stopping the processor's execution of code, as well as accessing memory while the device is stopped to download code. The Debug and Calibration Interface (DCI) provides the central run control including cross-triggering of breakpoint conditions. This DCI provides support for the IEEE 1149.1 JTAG standard with the JTAG Controller (JTAGC) as well as the IEEE 1149.7 standard with the Compact JTAG (CJTAG) module. The JTAG interface supports both Boundary Scan and Debug modes. It is based on the IEEE 1149.1 and IEEE 1149.7 standards. The JTAG Controller (JTAGC) operates in the standard IEEE 1149.1 5-wire interface. The JTAGC can either be fed directly from the JTAG pins or the CJTAG module, which supports the IEEE 1149.7 standard.

The Sequence Processing Unit (SPU) provides cross-triggering of events from the device. The SPU can access some of the DCU run control features since it can control the $\overline{\text{EVTO}}$ pins and monitor the EVTI pins. Furthermore, the SPU can use system triggers to cause

many different types of actions, including triggering a breakpoint cross-trigger (managed by the DCI).

Table 121. Reference links to related information

Related module	Reference
Production Device Debug and Calibration Interface (DCI)	Chapter 65: Debug and Calibration Interface (DCI)
Buddy Device Debug and Calibration Interface (DCI)	Chapter 92: Buddy Device—Debug and Calibration Interface (DCI)
JTAG Controller (JTAGC)	Chapter 66: JTAG Controller (JTAGC)
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)	Chapter 67: IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)
JTAG Data Communication (JDC)	Chapter 68: JTAG Data Communication (JDC)
Sequence Processing Unit (SPU)	Chapter 69: Sequence Processing Unit (SPU)

12.2.1 Debug and Calibration Interface (DCI)

The Debug and Calibration Interface (DCI) module provides debug and calibration features. The DCI module includes the device JTAG Controller and the IEEE 1149.7 interface. The DCI provides the following features:

- Debug mode enable control for connected tools or software
- Port-sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- IEEE 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control

Simultaneous operation for debug and calibration featuresThe DCI sends and receives the EVTI[1:0] and EVTO[1:0] signals to/from the cores and other modules. The tables below summarize the source and destination of EVTI, EVTO, and related signals for the PD and the BD. For example, for the PD, EVTI[0] are outputs of the JTAGM and SPU and inputs to the Cores, HSM, GTM, and NXMC.

Table 122. PD DCI signals

Signal	Cores	HSM	GTM	NXMC	NAR	JTAGM	SPU	DTS
EVTI[0]	I	I	I	I	—	O	I/O	—
EVTI[1]	—	—	—	—	—	O	I/O	—
EVTO[0]	O	O	O	O	—	I	I/O	—
EVTO[1]	—	—	—	—	—	I	I/O	O
Core debug event	I	—	—	—	—	—	—	—
DCI cross triggering	O	—	O	—	—	—	—	—

Table 123. BD DCI signals

Signal	RWA	NAR	NAL	JTAGM
$\overline{\text{EVTI}}[0]$	—	I	—	O
$\overline{\text{EVTI}}[1]$	—	—	—	O
$\overline{\text{EVTO}}[0]$	—	—	—	I
$\overline{\text{EVTO}}[1]$	—	—	—	I

The DCI sends the system debug signal, `ipg_debug`, to the following modules. For details regarding module functionality during system debug, refer to the individual module chapter.

- AMU
- CAN
- LFAST (Interprocessor Communications instance only)
- DSPI
- eDMA
- FCCU
- GTMINT
- I2C
- NXMC
- PIT
- PSI5
- SARADC
- SDADC
- SRX
- SSCM
- STM
- SWT

DCI $\overline{\text{EVTx}}$ Pin Multiplexing Control Register (DCI_PINCR)

The DCI $\overline{\text{EVTx}}$ pin multiplexing control register (DCI_PINCR) is described below. The $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ functions share the same set of possible pins. If this register is written to enable both $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ on the same pin, the $\overline{\text{EVTI}}$ function is selected, although this type of programming should be avoided. Programming the $\overline{\text{EVTI}}$ fields to select $\overline{\text{EVTI}}[0]$ or $\overline{\text{EVTI}}[1]$ inputs to come from more than one pin is considered a programming error and no pin is enabled.

Figure 60. DCI EVTx pin multiplexing control register (DCI_PINCR)

	31	30	29	28	27	26	25	24
R	Reserved							
W								
Reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
Reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	Reserved	EVTO1C	EVTO1B	EVTO1A	EVTO0D	EVTO0C	EVTO0B	EVTO0A
W								
Reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved	EVTI1C	EVTI1B	EVTI1A	EVTI0D	EVTI0C	EVTI0B	EVTI0A
W								
Reset	0	0	0	0	0	0	0	0

Table 124. DCI_PINCR register field descriptions

Field	Description
31-15	Reserved
14 EVTO1C	Enables $\overline{\text{EVTO}}[1]$ function on pin PM[6]
13 EVTO1B	Enables $\overline{\text{EVTO}}[1]$ function on pin PK[14]
12 EVTO1A	Enables $\overline{\text{EVTO}}[1]$ function on pin PH[11]
11 EVTO0D	Enables $\overline{\text{EVTO}}[0]$ function on pin PM[5]
10 EVTO0C	Enables $\overline{\text{EVTO}}[0]$ function on pin PM[4]
9 EVTO0B	Enables $\overline{\text{EVTO}}[0]$ function on pin PF[15]
8 EVTO0A	Enables $\overline{\text{EVTO}}[0]$ function on pin PF[14]

Table 124. DCI_PINCR register field descriptions (continued)

Field	Description
7	Reserved
6 EVTI1C	Enables $\overline{\text{EVTI}}[1]$ function on pin PM[6]
5 EVTI1B	Enables $\overline{\text{EVTI}}[1]$ function on pin PK[14]
4 EVTI1A	Enables $\overline{\text{EVTI}}[1]$ function on pin PH[11]
3 EVTI0D	Enables $\overline{\text{EVTI}}[0]$ function on pin PM[5]
2 EVTI0C	Enables $\overline{\text{EVTI}}[0]$ function on pin PM[4]
1 EVTI0B	Enables $\overline{\text{EVTI}}[0]$ function on pin PF[15]
0 EVTI0A	Enables $\overline{\text{EVTI}}[0]$ function on pin PF[14]

12.2.1.1 JTAG Controller (JTAGC)

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format. All Nexus debug registers are accessed and configured via the JTAG Controller.

12.2.1.2 Compact JTAG (CJTAG)

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities.

12.2.2 JTAG Data Communication (JDC)

The JTAG Data Communication (JDC) module allows both JTAG and software access to two registers. These two registers are the challenge/response password authorization registers used by the Hardware Security Module (HSM).

12.2.3 Sequence Processing Unit (SPU)

The Sequence Processing Unit (SPU) provides on-device logic analyzer trigger functions, such as performance monitoring, by using internal device signals as trigger sources. Performance monitor functions are available at both the system and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System-level performance monitor functions are integrated into the SPU complex trigger logic, thus allowing the counting and timing of any debug trigger combinations supported by the SPU.

Various clients generate watchpoints and other triggers when operating in Debug mode. The SPU collects these triggers and uses them as conditions for programmable sequences

of states and resultant actions. The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, it achieves complex logic-analyzer-like behavior, providing vital real-time visibility and the ability to debug system activities.

12.2.3.1 SPU Level1 Input Mux Configuration registers

The SPU Level1 Mux inputs for the L1SELn registers (where n = 0–7) are described in the following tables. The L1SEL0 register fields are described in the following table.

Table 125. L1SEL0 register field descriptions

Field	Description
MUX 0	000 CPU0 instruction address compare 1 (watchpoint 0) 001 CPU1 instruction address compare 1 (watchpoint 0) 010 CPU2 instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event input 1 (watchpoint 10) 111 CPU0 debug event output 2 (watchpoint 20)
MUX 1	000 CPU0 instruction address compare 2 (watchpoint 1) 001 CPU1 instruction address compare 2 (watchpoint 1) 010 CPU2 instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 CPU1 debug event input 1 (watchpoint 10) 111 CPU1 debug event output 2 (watchpoint 20)
MUX 2	000 CPU0 instruction address compare 3 (watchpoint 2) 001 CPU1 instruction address compare 3 (watchpoint 2) 010 CPU2 instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event input 1 (watchpoint 10) 111 CPU2 debug event output 2 (watchpoint 20)
MUX 3	000 CPU0 instruction address compare 4 (watchpoint 3) 001 CPU1 instruction address compare 4 (watchpoint 3) 010 CPU2 instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 4	000 CPU0 instruction address compare 5 (watchpoint 8) 001 CPU1 instruction address compare 5 (watchpoint 8) 010 CPU2 instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event input 2 (watchpoint 11) 111 CPU0 debug event output 3 (watchpoint 21)

Table 125. L1SEL0 register field descriptions (continued)

Field	Description
MUX 5	000 CPU0 instruction address compare 6 (watchpoint 9)
	001 CPU1 instruction address compare 6 (watchpoint 9)
	010 CPU2 instruction address compare 6 (watchpoint 9)
	011 Reserved
	100 Reserved
	101 Reserved
	110 CPU1 debug event input 2 (watchpoint 11)
	111 CPU1 debug event output 3 (watchpoint 21)
MUX 6	000 CPU0 instruction address compare 7 (watchpoint 14)
	001 CPU1 instruction address compare 7 (watchpoint 14)
	010 CPU2 instruction address compare 7 (watchpoint 14)
	011 Reserved
	100 Reserved
	101 Reserved
	110 CPU2 debug event input 2 (watchpoint 11)
	111 CPU2 debug event output 3 (watchpoint 21)
MUX 7	000 CPU0 instruction address compare 8 (watchpoint 15)
	001 CPU1 instruction address compare 8 (watchpoint 15)
	010 CPU2 instruction address compare 8 (watchpoint 15)
	011 Reserved
	100 Reserved
	101 Reserved
	110 Reserved
	111 Reserved

The L1SEL1 register fields are described in the following table.

Table 126. L1SEL1 register field descriptions

Field	Description
MUX 8	000 CPU0 instruction address compare 1 (watchpoint 0)
	001 CPU1 instruction address compare 1 (watchpoint 0)
	010 CPU2 instruction address compare 1 (watchpoint 0)
	011 Reserved
	100 Reserved
	101 Reserved
	110 CPU0 debug event output 0 (watchpoint 12)
	111 Reserved
MUX 9	000 CPU0 instruction address compare 2 (watchpoint 1)
	001 CPU1 instruction address compare 2 (watchpoint 1)
	010 CPU2 instruction address compare 2 (watchpoint 1)
	011 Reserved
	100 Reserved
	101 Reserved
	110 CPU1 debug event output 0 (watchpoint 12)
	111 Reserved

Table 126. L1SEL1 register field descriptions (continued)

Field	Description
MUX 10	000 CPU0 instruction address compare 3 (watchpoint 2) 001 CPU1 instruction address compare 3 (watchpoint 2) 010 CPU2 instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event output 0 (watchpoint 12) 111 Reserved
MUX 11	000 CPU0 instruction address compare 4 (watchpoint 3) 001 CPU1 instruction address compare 4 (watchpoint 3) 010 CPU2 instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 12	000 CPU0 instruction address compare 5 (watchpoint 8) 001 CPU1 instruction address compare 5 (watchpoint 8) 010 CPU2 instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event output 1 (watchpoint 13) 111 Reserved
MUX 13	000 CPU0 instruction address compare 6 (watchpoint 9) 001 CPU1 instruction address compare 6 (watchpoint 9) 010 CPU2 instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 CPU1 debug event output 1 (watchpoint 13) 111 Reserved
MUX 14	000 CPU0 instruction address compare 7 (watchpoint 14) 001 CPU1 instruction address compare 7 (watchpoint 14) 010 CPU2 instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event output 1 (watchpoint 13) 111 Reserved
MUX 15	000 CPU0 instruction address compare 8 (watchpoint 15) 001 CPU1 instruction address compare 8 (watchpoint 15) 010 CPU2 instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL2 register fields are described in the following table.

Table 127. L1SEL2 register field descriptions

Field	Description
MUX16	000 CPU0 data address compare 1 (watchpoint 4) 001 CPU1 data address compare 1 (watchpoint 4) 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 CPU0 write to L1CSR0 101 Reserved 110 Reserved 111 Reserved
MUX 17	000 CPU0 data address compare 2 (watchpoint 5) 001 CPU1 data address compare 2 (watchpoint 5) 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 CPU0 write to L1CSR1 101 Reserved 110 Reserved 111 Reserved
MUX 18	000 CPU0 data address compare 3 (watchpoint 6) 001 CPU1 data address compare 3 (watchpoint 6) 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 CPU0 write to E2ECTL0 101 Reserved 110 Reserved 111 Reserved
MUX 19	000 CPU0 data address compare 4 (watchpoint 7) 001 CPU1 data address compare 4 (watchpoint 7) 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 CPU0 write to DMEMCTL0 101 Reserved 110 Reserved 111 Reserved
MUX 20	000 CPU0 data address compare 1 (watchpoint 4) 001 CPU1 data address compare 1 (watchpoint 4) 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 CPU0 write to IMEMCTL0 101 Reserved 110 CPU2 write to L1CSR0 111 Reserved
MUX 21	000 CPU0 data address compare 2 (watchpoint 5) 001 CPU1 data address compare 2 (watchpoint 5) 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 CPU0 write to HID0 101 CPU1 write to MSR 110 CPU2 write to L1CSR1 111 Reserved

Table 127. L1SEL2 register field descriptions (continued)

Field	Description
MUX 22	000 CPU0 data address compare 3 (watchpoint 6)
	001 CPU1 data address compare 3 (watchpoint 6)
	010 CPU2 data address compare 3 (watchpoint 6)
	011 Reserved
	100 CPU0 write to HID1
	101 CPU1 write to IVPR
	110 CPU2 write to E2ECTL0
	111 Reserved
MUX 23	000 CPU0 data address compare 4 (watchpoint 7)
	001 CPU1 data address compare 4 (watchpoint 7)
	010 CPU2 data address compare 4 (watchpoint 7)
	011 Reserved
	100 CPU0 write to BUCSR
	101 CPU1 write to SPEFSCR
	110 CPU2 write to DMEMCTL0
	111 Reserved

The L1SEL3 register fields are described in the following table.

Table 128. L1SEL3 register field descriptions

Field	Description
MUX 24	000 CPU0 data address compare 1 (watchpoint 4)
	001 CPU1 data address compare 1 (watchpoint 4)
	010 CPU2 data address compare 1 (watchpoint 4)
	011 Reserved
	100 CPU0 write to SPEFSCR
	101 CPU1 write to BUCSR
	110 CPU2 write to IMEMCTL0
	111 Reserved
MUX 25	000 CPU0 data address compare 2 (watchpoint 5)
	001 CPU1 data address compare 2 (watchpoint 5)
	010 CPU2 data address compare 2 (watchpoint 5)
	011 Reserved
	100 CPU0 write to IVPR
	101 CPU1 write to HID1
	110 CPU2 write to HID0
	111 Reserved
MUX 26	000 CPU0 data address compare 3 (watchpoint 6)
	001 CPU1 data address compare 3 (watchpoint 6)
	010 CPU2 data address compare 3 (watchpoint 6)
	011 Reserved
	100 CPU0 write to MSR
	101 CPU1 write to HID0
	110 CPU2 write to HID1
	111 Reserved

Table 128. L1SEL3 register field descriptions (continued)

Field	Description
MUX 27	000 CPU0 data address compare 4 (watchpoint 7) 001 CPU1 data address compare 4 (watchpoint 7) 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 CPU1 write to IMEMCTL0 110 CPU2 write to BUCSR 111 Reserved
MUX 28	000 CPU0 data address compare 1 (watchpoint 4) 001 CPU1 data address compare 1 (watchpoint 4) 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 CPU1 write to DMEMCTL0 110 CPU2 write to SPEFSCR 111 Reserved
MUX 29	000 CPU0 data address compare 2 (watchpoint 5) 001 CPU1 data address compare 2 (watchpoint 5) 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 CPU1 write to E2ECTL0 110 CPU2 write to IVPR 111 Reserved
MUX 30	000 CPU0 data address compare 3 (watchpoint 6) 001 CPU1 data address compare 3 (watchpoint 6) 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 CPU1 write to L1CSR1 110 CPU2 write to MSR 111 Reserved
MUX 31	000 CPU0 data address compare 4 (watchpoint 7) 001 CPU1 data address compare 4 (watchpoint 7) 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 CPU1 write to L1CSR0 110 Reserved 111 Reserved

The L1SEL4 register fields are described in the following table.

Table 129. L1SEL4 register field descriptions

Field	Description
MUX 32	000 CPU0 performance monitor counter 0 (watchpoint 23) 001 CPU1 performance monitor counter 0 (watchpoint 23) 010 CPU0 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Production device NAR full 111 GTM watchpoint 0
MUX 33	000 CPU0 performance monitor counter 1 (watchpoint 24) 001 CPU1 performance monitor counter 1 (watchpoint 24) 010 CPU1 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Production device NAR partial full 111 GTM watchpoint 1
MUX 34	000 CPU0 performance monitor counter 2 (watchpoint 25) 001 CPU1 performance monitor counter 2 (watchpoint 25) 010 CPU2 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Buddy device NAR partial full 111 GTM watchpoint 2
MUX 35	000 CPU0 performance monitor counter 3 (watchpoint 26) 001 CPU1 performance monitor counter 3 (watchpoint 26) 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Buddy device NAR full 111 GTM watchpoint 3
MUX 36	000 CPU2 performance monitor counter 0 (watchpoint 23) 001 Reserved 010 CPU0 performance monitor event input (watchpoint 22) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 4
MUX 37	000 CPU2 performance monitor counter 1 (watchpoint 24) 001 Reserved 010 CPU1 performance monitor event input (watchpoint 22) 011 Reserved 100 NXMC1 watchpoint 7 101 Reserved 110 Reserved 111 GTM watchpoint 5

Table 129. L1SEL4 register field descriptions (continued)

Field	Description
MUX 38	000 CPU2 performance monitor counter 2 (watchpoint 25)
	001 Reserved
	010 CPU2 performance monitor event input (watchpoint 22)
	011 Reserved
	100 Reserved
	101 Reserved
	110 Reserved
	111 GTM watchpoint 6
MUX 39	000 CPU2 performance monitor counter 3 (watchpoint 26)
	001 Reserved
	010 Reserved
	011 Reserved
	100 Reserved
	101 Reserved
	110 Reserved
	111 GTM watchpoint 7

The L1SEL5 register fields are described in the following table.

Table 130. L1SEL5 register field descriptions

Field	Description
MUX 40	000 CPU0 interrupt (watchpoint 16)
	001 CPU0 critical interrupt (watchpoint 18)
	010 CPU0 data trace control range 1 (watchpoint 29)
	011 Reserved
	100 Reserved
	101 Reserved
	110 SPU counter event 0
	111 GTM watchpoint 8
MUX 41	000 CPU1 interrupt (watchpoint 16)
	001 CPU1 critical interrupt (watchpoint 18)
	010 CPU1 data trace control range 1 (watchpoint 29)
	011 Reserved
	100 Reserved
	101 Reserved
	110 SPU counter event 1
	111 GTM watchpoint 9
MUX 42	000 CPU2 interrupt (watchpoint 16)
	001 CPU2 critical interrupt (watchpoint 18)
	010 CPU2 data trace control range 1 (watchpoint 29)
	011 Reserved
	100 Reserved
	101 Reserved
	110 SPU counter event 2
	111 GTM watchpoint 10

Table 130. L1SEL5 register field descriptions (continued)

Field	Description
MUX43	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 3 111 GTM watchpoint 11
MUX 44	000 CPU0 return (watchpoint 17) 001 CPU0 critical return (watchpoint 19) 010 CPU0 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 4 111 Reserved
MUX 45	000 CPU1 return (watchpoint 17) 001 CPU1 critical return (watchpoint 19) 010 CPU1 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 5 111 Reserved
MUX 46	000 CPU2 return (watchpoint 17) 001 CPU2 critical return (watchpoint 19) 010 CPU2 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 6 111 Reserved
MUX 47	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 7 111 Reserved

The L1SEL6 register fields are described in the following table.

Table 131. L1SEL6 register field descriptions

Field	Description
MUX 48	000 CPU0 context load (using e_lmvspw instruction) 001 CPU0 context restore (using e_stmvspw instruction) 010 CPU0 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 0 100 Reserved 101 Reserved 110 SPU counter event 8 111 Reserved
MUX 49	000 CPU1 context load (using e_lmvspw instruction) 001 CPU1 context restore (using e_stmvspw instruction) 010 CPU1 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 1 100 Reserved 101 Reserved 110 SPU counter event 9 111 Reserved
MUX 50	000 CPU2 context load (using e_lmvspw instruction) 001 CPU2 context restore (using e_stmvspw instruction) 010 CPU2 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 2 100 Reserved 101 Reserved 110 SPU counter event 10 111 Reserved
MUX 51	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 3 100 Reserved 101 Reserved 110 SPU counter event 11 111 Reserved
MUX 52	000 CPU0 interrupt current priority match to C0PIS 001 CPU0 write to ME bits of MSR 010 Reserved 011 NXMC0 watchpoint 4 100 Reserved 101 Reserved 110 SPU counter event 12 111 Reserved
MUX 53	000 CPU1 interrupt current priority match to C1PIS 001 CPU1 write to ME bits of MSR 010 Reserved 011 NXMC0 watchpoint 5 100 Reserved 101 Reserved 110 SPU counter event 13 111 Reserved

Table 131. L1SEL6 register field descriptions (continued)

Field	Description
MUX 54	000 CPU2 interrupt current priority match to C2PIS
	001 CPU2 write to ME bits of MSR
	010 Reserved
	011 NXMC0 watchpoint 6
	100 Reserved
	101 Reserved
	110 SPU counter event 14
	111 Reserved
MUX 55	000 Reserved
	001 Reserved
	010 Reserved
	011 NXMC0 watchpoint 7
	100 Reserved
	101 Reserved
	110 SPU counter event 15
	111 Reserved

The L1SEL7 register fields are described in the following table.

Table 132. L1SEL7 register field descriptions

Field	Description
MUX 56	000 CPU0 write to EE bits of MSR
	001 CPU0 write to CE bits of MSR
	010 NXMC1 watchpoint 0
	011 Reserved
	100 Reserved
	101 Reserved
	110 Reserved
	111 DCI EVTIO
MUX 57	000 CPU1 write to EE bits of MSR
	001 CPU1 write to CE bits of MSR
	010 NXMC1 watchpoint 1
	011 Reserved
	100 Reserved
	101 Reserved
	110 Reserved
	111 DCI EVTI1
MUX 58	000 CPU2 write to EE bits of MSR
	001 CPU2 write to CE bits of MSR
	010 NXMC1 watchpoint2
	011 Reserved
	100 Reserved
	101 Reserved
	110 Reserved
	111 DCI EVTO0

Table 132. L1SEL7 register field descriptions (continued)

Field	Description
MUX 59	000 Reserved 001 Reserved 010 NXMC1 watchpoint 3 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO1
MUX 60	000 CPU0 processor exception vector match to C0PEVP 001 CPU0 write to DE bits of MSR 010 NXMC1 watchpoint 4 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI system halt
MUX 61	000 CPU1 processor exception vector match to C1PEVP 001 CPU1 write to DE bits of MSR 010 NXMC1 watchpoint 5 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 62	000 CPU2 processor exception vector match to C2PEVP 001 CPU2 write to DE bits of MSR 010 NXMC1 watchpoint 6 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 63	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

12.3 Calibration interface

This device includes an alternate calibration interface for high-speed run control and advanced software control. This calibration interface is based on the LFAST interface. The LFAST based calibration has the ability to perform all of the functions that JTAG supports, only at a faster speed. The MCU implements the slave module, and the pins for the LFAST interface are multiplexed onto the same pins as the JTAG pins. Initially, the device powers up with the JTAG module in control of the pins. Under tool commands, the LFAST interface

can be enabled and the pins become LFAST signals. Once the LFAST interface has been activated, all run control and other JTAG operations are converted in the MCU from LFAST messages into JTAG control operations of the JTAG Controller. The interface is supported entirely in hardware for both LFAST and JTAG modes, with no software or system overhead required to support interface traffic.

Table 133. Reference links to related information

Related module	Reference
JTAG Master (JTAGM)	Chapter 70: JTAG Master (JTAGM)
LFAST Module—JTAGM	Chapter 71: LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed Debug
Development Tool Semaphore (DTS)	Chapter 72: Development Trigger Semaphore (DTS)

12.3.1 JTAG Master (JTAGM)

The JTAG Master (JTAGM) acts as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or through software.

The data transferred to this module is transformed to produce TCK, TMS, TDI, and TRST outputs and to accept TDO inputs. The module allows these five signals to connect to the JTAGC as if the JTAG data were coming from an external tool. The JTAGM generates all required JTAG scan chains to allow software and high-speed serial communication access to all JTAG-mapped resources.

12.3.2 Debug LFAST

The high-speed calibration interface provides up to a 320 Mbit/s interface between a calibration tool and the device. The calibration interface uses the LFAST as the physical layer between the tool and the JTAGM for access to debug resources. In addition, an LFAST switch allows direct access to read and write memory. The calibration debug LFAST module is one of two instantiations of the LFAST module on the device. The calibration LFAST is configured for slave-only operation and is meant to be used as a calibration interface to provide higher (than JTAG) speed debug and calibration operations. The calibration LFAST interface supports full duplex operation, where the available bandwidths for upstream and downstream traffic are independent.

12.3.3 Development Trigger Semaphore (DTS)

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways that this module can be used, including as a component of an external real-time data acquisition system.

Note: *When used as a component of a triggered data-acquisition system, Nexus read/write access is via the JTAG interface of the Nexus debug port and is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus auxiliary port.*

12.4 Debug over CAN

As well as supporting debug via JTAG and LFAST interfaces, the device supports the use of a CAN interface for debug. This allows debug of application hardware where access to the dedicated JTAG/LFAST interface signals is not easily available. The debug over CAN mechanism is intended to provide basic debug functionality with some reduction in bandwidth and features compared to use of the dedicated JTAG or LFAST interfaces.

Use of the CAN interface for debug purposes requires the use of some M_CAN, eDMA and JTAGM resources. Some initialization of these resources via software is required, but once this initialization has completed, debug over CAN is possible without any further software overhead.

As the debug over CAN scheme generates internal JTAG messages based on received CAN data, all JTAG clients and included debug resources are accessible. Basic device trace capability is also possible by configuring the device trace hardware to stream to a device overlay/trace RAM, which can be read later using debug over CAN.

The debug over CAN scheme supports the following features:

- Supports operation through M_CAN_0 or M_CAN_1 interfaces
 - Makes use of M_CAN debug enhancements
 - Debug message selected by filter configuration SFEC/EFEC=111
- Allows debug of hardware where JTAG access is not available
- Debug traffic carried over application CAN bus
- Debug traffic on CAN coexists with application traffic
- No software overhead after initialization
- flexible selection of CAN identifiers for debug use
- Access to all JTAG debug facilities, includes CPU run control

12.5 Nexus Trace Aurora interface

The Aurora interface allows Nexus protocol information to be transmitted serially at high speed over multiple Aurora lanes to provide advanced trace information from the device. The Aurora protocol handles the encoding of the data and the striping of information across the lanes. The Aurora interface is only available on the Emulation and Debug Device. Trace information is transferred to the Buddy Device for transmission via the Aurora interface.

The Aurora information can be used to reconstruct events or operations that occurred inside the microcontroller. Nexus supports the transmission, through a single Aurora port, of information from multiple trace clients within the MCU. Each Nexus message is tagged with the source client identifier and the type of message. The actual trace information that is available depends on the Nexus client; generally, it can be individually enabled (both the type of message and the client). [Table 134](#) shows a few examples of Nexus messages. All messages can be enabled to include a timestamp.

Table 134. Nexus message types

Message Type	Description
Program trace	Nexus program trace messages transmit any discontinuities in the program flow, such as branches and interrupts. Multiple types of messages can be generated.
Ownership trace	Ownership trace provides information on process identification changes.

Table 134. Nexus message types (continued)

Message Type	Description
Watchpoint trace	Watchpoint messages are generated any time a watchpoint match occurs in the program or data flow, or as a result of SPU actions. These can be programmed for many types of events within the MCU.
Device identification	The device identification message allows information about the MCU to be transmitted upon startup to allow tools to identify the target system MCU type.
Debug status	Debug status messages provide additional information that may be needed to reconstruct software execution, such as whether the device has entered Debug or Low-power mode.
Data acquisition	Data acquisition messages are an optional feature that allows software control of information to be transmitted.
error	Error messages are transmitted when an error condition occurs, such as internal buffer overflows.

The Nexus trace information can be output to internal memory, or the parallel information can be serialized for transmission over a serial interface through the Nexus Aurora interface. Nexus trace information can also be routed through the Data Write Processing Unit (DWPU) as described in [Section 12.7, Data trace filtering](#).

The Nexus blocks do not have memory-mapped registers. The Nexus registers are accessed by the development tool via the JTAG port using a two step process. First, the specific block is selected by loading the corresponding ACCESS_AUX instruction as described in the JTAGC chapter. Next, after the block is selected, the Nexus registers are enabled by loading the enable instruction to the JTAGC Instruction Register (IR). The enable instructions for the different Nexus clients are shown in the following table. After the enable instruction is received, the development tool has access to the Nexus registers of the selected client.

Table 135. Nexus client JTAG enable instructions

Module	Enable instruction	Opcode
Production Die		
Cores	CORE_ENABLE	0x7C (10 bits)
NXMC	NEXUS_ENABLE	0x0 (4 bits)
GTM		
NAR		
SPU		
Buddy Die		
RWA	CORE_ENABLE	0x7C (10 bits)
NAR	NEXUS_ENABLE	0x0 (4 bits)
NAL		
DWPU		

All of the internal JTAG modules are accessed through a single JTAG connection to the device. In the case of the ED, all JTAG communication is performed through the JTAG

interface located in the BD. In the production package configuration, access is performed through the JTAG interface located in the PD.

With the hierarchical JTAG configuration, the BD JTAGC is considered the master JTAG TAP controller. The PD JTAGC TAP controller is selected via an AUX_ACCESS instruction. Other TAP controllers residing on the BD are also selected via AUX_ACCESS instructions (NAR, RWA, BD_DCI). Once control is transferred from the BD JTAGC to the PD JTAGC via the AUX_ACCESS instruction, all other PD TAP controllers are then selected via AUX_ACCESS instructions executed by the PD JTAGC. This is also referenced as TAP-sharing and is shown in the following figure.

Note: *On the Emulation device, when the PAUSE-DR/UPDATE-DR state is executed to pass control of the JTAG TAP controller, control is always passed to the top level JTAG TAP controller, in other words, the DCI of the BD. For this reason, it is recommended that all JTAG operations, except when the BD clients need to be accessed, that all PD JTAG access be preceded with ACCESS_PD_JTAGC command. If the device is not an ED, the access is ignored.*

Figure 61. JTAG hierarchal view

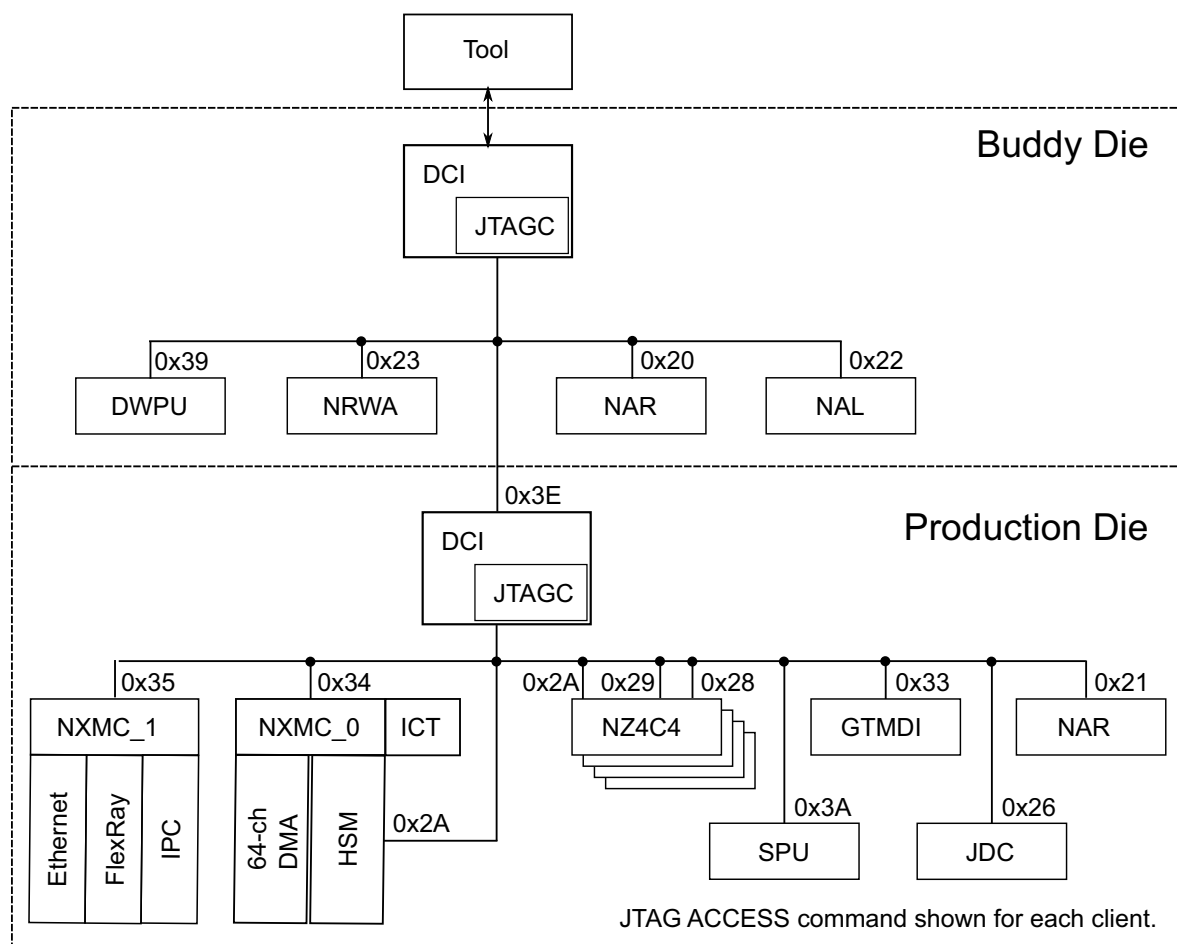


Table 136. Reference links to related information

Related module	Reference
Production Device Nexus Aurora Router (NAR)	Chapter 73: Nexus Aurora Router (NAR)
Buddy Device Nexus Aurora Router (NAR)	Chapter 95: Buddy Device—Nexus Aurora Router (NAR)
Buddy Device—Nexus Aurora Link (NAL)	Chapter 96: Buddy Device—Nexus Aurora Link (NAL)
Buddy Device—Nexus Aurora PHY (NAP)	Chapter 97: Buddy Device—Nexus Aurora PHY (NAP)
Buddy Device—Nexus Read Write Access (RWA)	Chapter 99: Buddy Device—Nexus Read Write Access (RWA)

12.5.1 Nexus Aurora overview

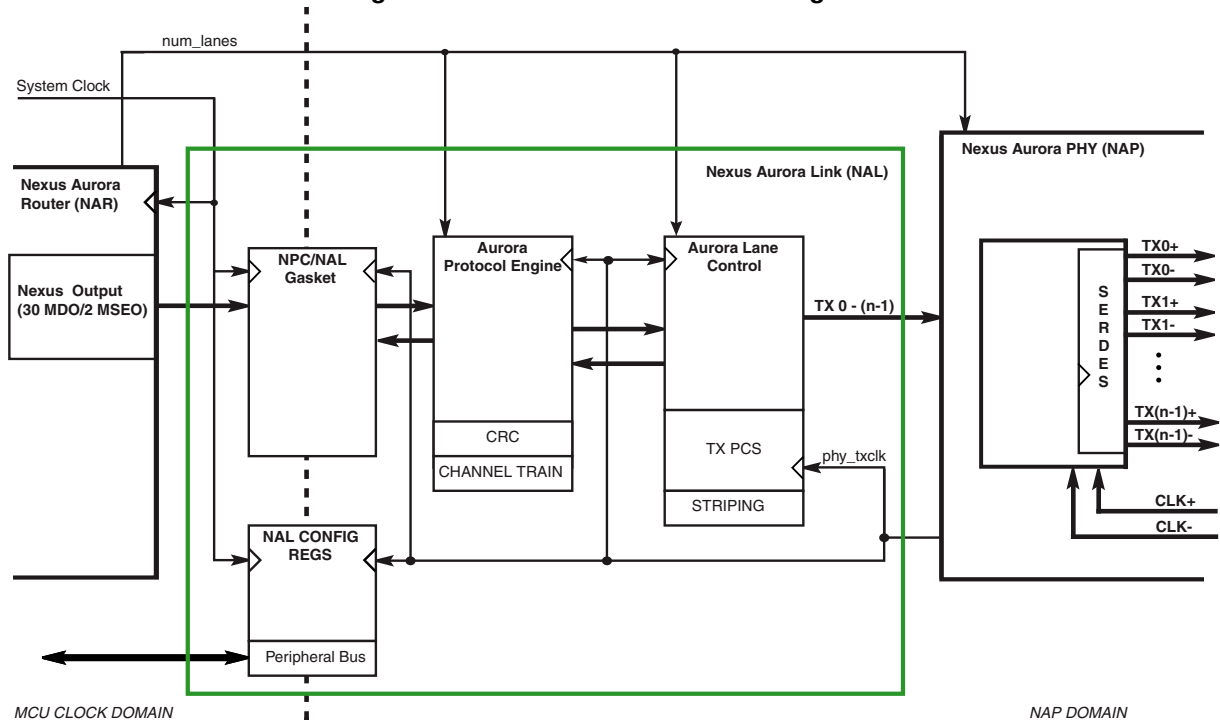
The IEEE-ISTO 5001-2011 standard adds a new type of auxiliary trace port that supports a higher bandwidth. This new interface type uses the Aurora protocol to reformat the parallel Nexus messages into one or more output serial lanes of data, where a lane is defined as a 2-pin Low Voltage Differential Signals (LVDS) interface. The protocol handles striping the data onto the multiple lanes. Each lane uses an 8b/10b encoded LVDS driver. The encoding allows the receiver to recover clocking information.

The Aurora physical interface is based on the XAUI (10 Gigabit Attachment Unit Interface) standard used in communication protocols, which covers the jitter, differential skew, inter-lane skew, and bit error rate requirements.

The Aurora protocol covers lane initialization and channel bonding for both simplex and duplex implementations. The current implementation uses a simplex output-only connection from the MCU to the external tool.

The Aurora interface connects Nexus parallel trace output to an Aurora Protocol Engine (APE), the Aurora Lane Control (ALC), and the Nexus Aurora Physical interface (NAP). [Figure 62](#) shows the Aurora interface block diagram, where the number of lanes, n , is 4.

Figure 62. Aurora interface block diagram

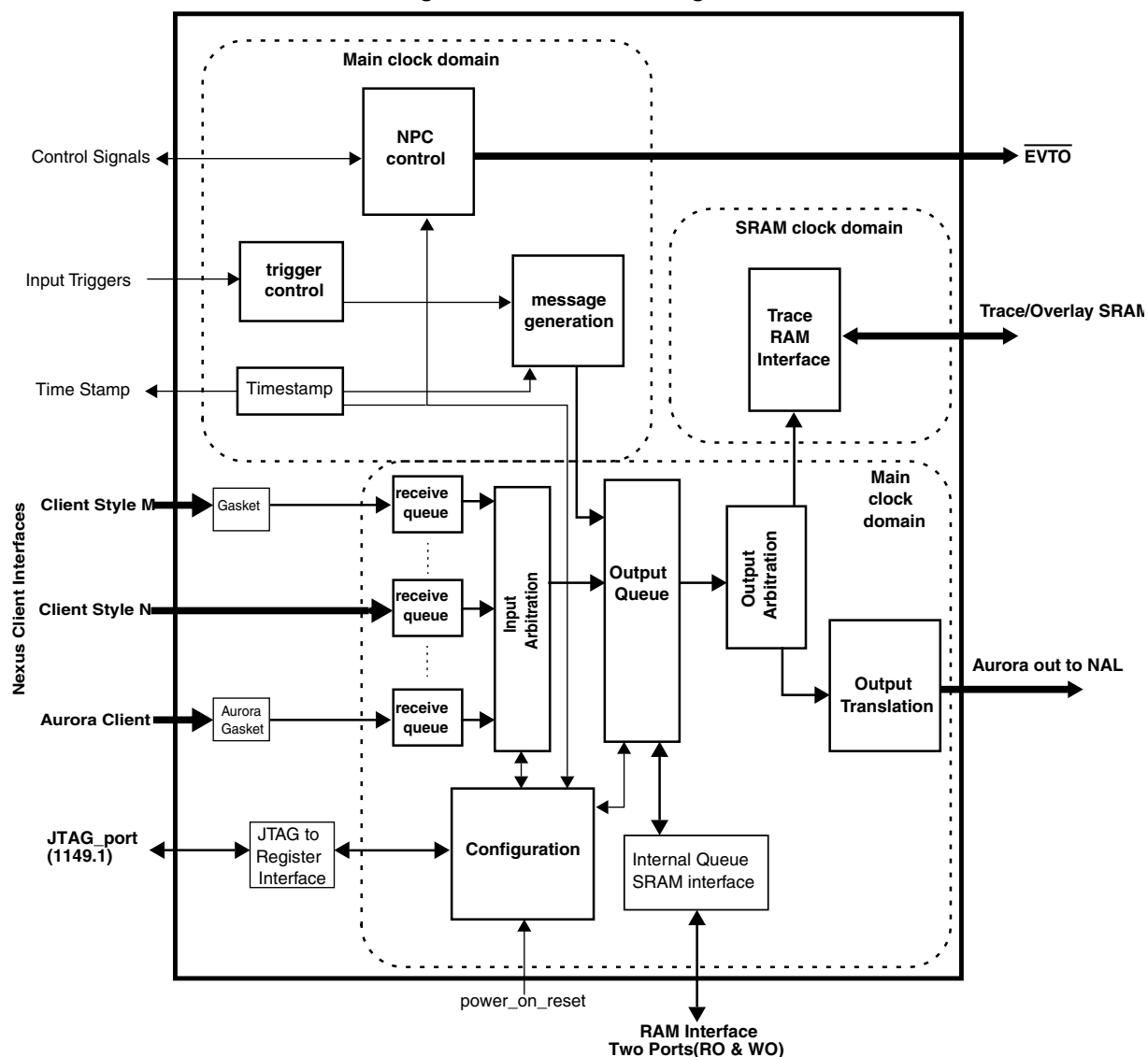


12.5.2 Nexus Aurora Router (NAR)

The Nexus Aurora Router (NAR) manages the Nexus trace information from each of the Nexus clients. The NAR takes the parallel trace information from each of the clients, buffers it to avoid overruns, and routes it to either the internal overlay RAM or the Buddy Device, where it is stored in on-chip SRAM or transmitted out of the device through the Aurora interface.

16 KB of trace SRAM is available on the PD (located in the flash module), and 1 MB of trace SRAM is available in the ED. (Some of this SRAM may be used for calibration overlay content as well as trace content.) Similarly, a smaller NAR output queue is located on the PD (2 KB), and a much larger NAR output queue is located in the ED (8 KB). The physical Aurora pins are only available on the ED. [Figure 63](#) shows the NAR block diagram.

Figure 63. NAR block diagram



The PD NAR module has seven clients, with client index 1–7. Clients 8–16 are not connected. The client connections and the corresponding NAR client index are shown in the following table. The NAR client index is used in the NAR_STCR, NAR_CSSR, and NAR_CDR registers. The source trace ID is used in the NAR_SFR.

Table 137. PD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
—	NAR	0b1111 (0xF)	—
1	GTM	0b0111 (0x7)	3
2	HSM	0b0011 (0x3)	1
3	Core 0	0b0000 (0x0)	1
4	NXMC Client 0 Concentrator	0b1100 (0xC)	1

Table 137. PD NAR clients (continued)

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
5	Core 1	0b0001 (0x1)	1
6	NXMC Client 1 Concentrator	0b1101 (0xD)	1
7	Core 2	0b0010 (0x2)	1
8	—	—	—
9	—	—	—
10	—	—	—
11	—	—	—
12	—	—	—
13	—	—	—
14	—	—	—
15	—	—	—
16	—	—	—

The BD NAR module has one client and the client connections and the corresponding NAR client index are shown in the following table.

Table 138. BD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
1	PD NAR	0b1110 (0xE)	2

The NAR is limited writing trace information to the overlay RAM. The PD NAR only has access to the 16 KB internal overlay RAM. Therefore, the PD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0D000000–0x0D003FFF. The BD NAR also has access to 1 MB of extended overlay RAM. Therefore, the BD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0C000000–0x0C1FFFFFF.

The NUM_LANES field of the NAR_NAPCR is described in the following table.

Table 139. NAR_NAPCR field descriptions

Field	Description
19–16 NUM_LANE	TX Lane Configuration. Number of enabled TX lanes. 0000 Disable NAP and NAL 0001 Reserved 0010 Enable 2 lanes (Lane 0–1) 0011 Reserved 0100 Enable 4 lanes (Lanes 0–3) 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001–1111 Reserved

12.5.3 Nexus Aurora Link (NAL)

The Nexus Aurora Link (NAL) is only included in the Emulation and Debug Device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer serializer and transmitter (no receiver). It includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol.

Debug data in the Nexus format from multiple clients is packetized into a 32-bit format by the clients and then collected by the NAR. The NAL encodes the message data (8b/10b), stripes it across the lanes, and inserts Aurora control characters (such as start of message, end of message, idle, clock compensation characters, and so on) around the payload. The NAL PCS drives the Nexus Aurora Physical (NAP) interface.

12.5.4 Nexus Aurora PHY (NAP)

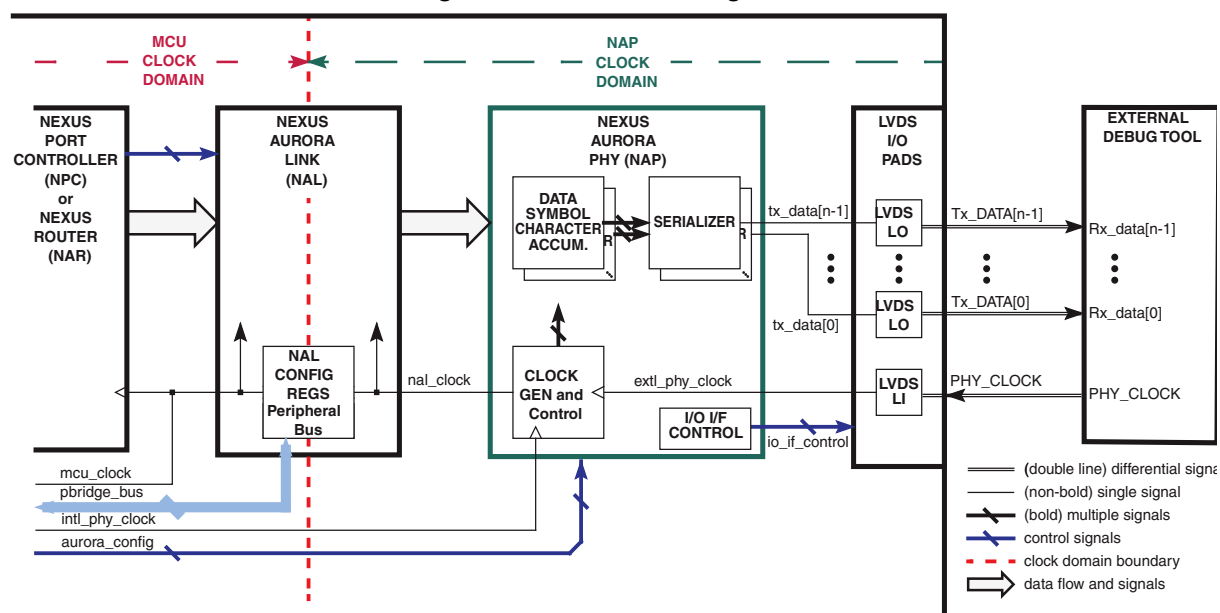
The Nexus Aurora PHY (NAP) is only included in the Emulation and Debug Device. The NAP, in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. The NAP receives 8b/10b encoded data from the ALC block in the NAL and transmits this data onto a given PHY lane's LVDS pads. The NAP performs lane striping with settings in the NAL and NAR specifying lane configuration. Each NAP lane consists of a Data Symbol Character Accumulator that registers the 10-bit character for that lane from the ALC on the rising edge of the NAL clock. A serializer takes the 10-bit character and serializes it into the individual transmit data bits at the PHY clock rate to be transmitted via the LVDS output drivers to the external debugger receive channel.

Note: The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive data side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called "lane alignment" in Aurora terminology), 10b/8b decoding, decode and disparity error tracking, and CRC checking in the received data stream.

Figure 64 shows a block diagram of the Nexus Aurora Physical (NAP) interface and the LVDS output drivers, where the number of lanes, *n*, is 4.

Figure 64. NAP block diagram



12.6 Nexus clients

Nexus supports the transmission, through a single Aurora port, of information from multiple trace clients within the device.

Table 140. Reference links to related information

Related module	Reference
e200z420n3 Nexus Support	Chapter 74: e200z420n3 Nexus 3 Module
e200z425n3 Nexus Support	Chapter 75: e200z425n3 Nexus 3 Module
GTM Development Interface (GTMDI)	Chapter 76: GTM Development Interface (GTMDI)
Nexus Crossbar Multi-Master Client (NXMC)	Chapter 77: Nexus Crossbar Multi-Master Client (NXMC)

12.6.1 e200z4 Nexus 3

The e200z420n3 and e200z425n3 Nexus 3 modules provide real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering and Processor Overrun Control.

12.6.2 GTM Development Interface (GTMDI)

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including Multi-channel Sequencer (MCS) cores, Advanced Routing Unit (ARU) data trace, Input Channel Filters trace, Output Timer channels, DPLL and Sensor Pattern Evaluation (SPE) module submodules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules to implement complex debug features. Trace capability is also provided along with a global timestamp that allows real-time trace for several GTM submodule events. The GTMDI implements a Nexus client standard interface and a JTAG standard port controller to provide communication capability via the JTAG port and the on-chip NAR.

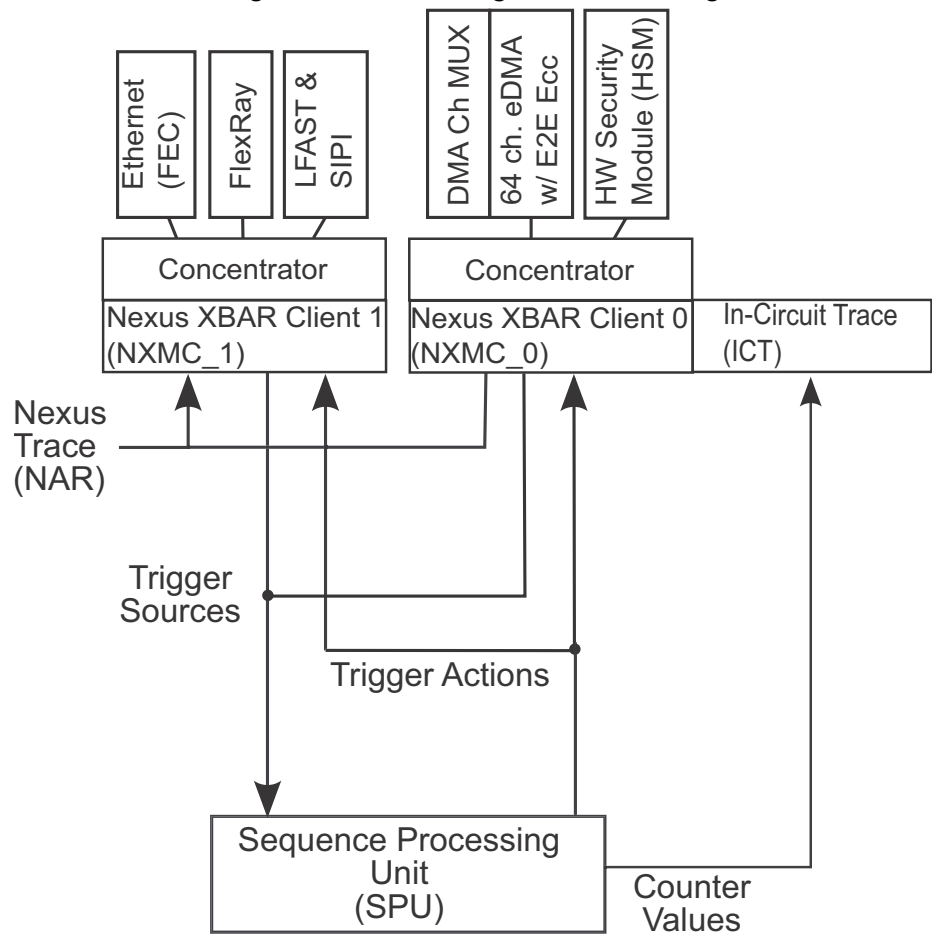
12.6.3 Nexus Crossbar Multi-master Client (NXMC)

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2011 standard. This module provides development support capabilities for SoCs without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard.

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Two NXMCs are implemented: one for the fast masters and one for the slow masters. Trace messages transmitted over the Nexus interface (or stored in trace memory) include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message. In addition to XBAR bus master trace capability, the NXMC also allows for in-circuit trace capability that can be used to trace the Sequence Processing Unit (SPU) counter values. The two external hardware watchpoint triggers of each NXMC are used to generate timestamps (four total—two from each NXMC) for events from the SPU.

The following figure shows a block diagram illustrating the NXMC integration.

Figure 65. NXMC integration block diagram



The values of the trace master ID or in-circuit client ID, along with the Nexus message source identifier for the different clients are shown in the following table.

Table 141. NXMC source and master client IDs

Client		Nexus trace source ID (SRC)	Trace master ID or In-circuit trace client ID (MASTER)
In-Circuit Trace	SPU	0b1011	0b0000
XBAR Client 0 Concentrator (fast masters)	Hardware Security Module (HSM)	0b1100	0b0000
	eDMA Controller		0b0001
	Reserved		0b0010–0b0111

Table 141. NXMC source and master client IDs (continued)

Client		Nexus trace source ID (SRC)	Trace master ID or In-circuit trace client ID (MASTER)
XBAR Client 1 Concentrator (slow masters)	System Bus	0b1101	0b1000
	FlexRay		0b1001
	Ethernet		0b1010
	Reserved		0b1011–0b1110
Data Write Processor Unit (DWPU)	e200zx core	— ⁽¹⁾	0b1111

1. The Nexus trace Source ID is the same as the original core SRC.

12.7 Data trace filtering

The Data Write Processing Unit (DWPU) supports data trace filtering as well as out of range parameter identification.

Table 142. Reference links to related information

Related module	Reference
Data Write Processing Unit (DWPU)	Chapter 98: Buddy Device—Data Write Processing Unit (DWPU)

DWPU operates in conjunction with the BD NAR and a small RAM array. DWPU receives its input trace stream from the BD NAR Nexus Out Link. The trace stream is initially decompressed (which includes conversion of relative address and data values to absolute values) to ease subsequent filtering and comparison operations. The decompression process tracks multiple contexts, with individual context tracking required for each supported Nexus client.

The processed stream then passes through an address compare block, that generates tag RAM addresses corresponding to a portion of each data write address within the trace stream. Another portion of the data write addresses is used to select a particular bit from the data word returned by the tag RAM, so that each tag RAM bit corresponds to a unique 64-bit data write address. The value of the selected bit determines whether the data write trace occurrence is passed out of the DWPU back to the BD NAR, or is removed from the trace stream. DWPU returns the data trace filtered and re-formatted stream back to a secondary Nexus client input port on the BD NAR.

13 Core Complex Overview

The SPC57EM80xx has four separate cores that perform various computational and control functions. The main computational shell consists of Main Core_0 and Main Core_1, both of which use an e200z420n3 core. These two cores are used for general computational functions. A third core, referred to as Checker Core_0s, uses an e200z419 core, which is a true subset of the e200z420n3 core. When enabled, Checker Core_0s operates in Lock Step mode with Main Core_0 executing exactly the same instructions as Main Core_0. Checker Core_0s checks to ensure that Main Core_0 is executing correctly. There is no Lock Step function associated with Main Core_1. The e200z419 core, used as Checker Core_0s, differs in hardware from the e200z420n3 in that it has no Instruction Cache, Data Cache, I-memory, or D-memory. The fourth processor, Peripheral Core_2, employs an e200z425n3 core and is generally used to control various I/O modules. The register set for these three cores is identical.

Note: The processor ID for each CPU core is contained in its Processor ID Register (PIR).

This chapter provides overviews of the e200z420n3 core, the e200z419 core and the e200z425n3 core.

13.1 Overview of the e200z420n3, e200z419, and e200z425n3 cores

The e200z processor family is a set of CPU cores that implement low-cost versions of the Power Architecture.

Each core integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The e200z420n3, e200z419, and e200z425n3 are dual-issue, 32-bit Power VLE (variable-length-encoding) compliant designs with 32-bit general-purpose registers (GPRs) for improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document. The base *PowerISA 2.06* fixed-length 32-bit instruction set is not directly supported.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision embedded numeric operations using the general-purpose registers.

13.2 Features

The following is a list of some of the key features of the e200z420n3, e200z419, and e200z425n3 processor cores:

- Dual issue, 32-bit *PowerISA 2.06*-VLE compliant CPU
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
 - Dedicated branch address calculation adder
 - Branch target prefetching using BTB allows for single cycle branches in many cases
- Load/store unit
 - 2 cycle load latency
 - Fully pipelined
 - Misaligned access support
- 32-bit General-Purpose Register (GPR) file
- Dual AHB 2.v6 64-bit system buses
- Local Instruction and Data Memories (IMEM, DMEM) with shared AHB 2.v6 64-bit slave interface
 - e200z420n3 core—16 KB IMEM, 64 KB DMEM
 - e200z425n3 core—16 KB IMEM, 64 KB DMEM
 - e200z419 core does not contain either Instruction or Data Memories
- Memory Protection Unit (MPU) implementing a 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized instruction or data memory regions
- 2-Way Set Associative Harvard Instruction and Data Cache
 - e200z420n3 core—8 KB ICache, 4 KB DCache
 - e200z425n3 core—8 KB ICache, no DCache
 - e200z419 core does not contain either Instruction or Data Cache
- Embedded Floating-point APU (EFPU2) supporting single-precision floating-point operations
- Lightweight Signal Processing (LSP) APU (e200z425n3 core only) supporting fixed-point multiply/accumulate and integer/fixed-point operations using the 32-bit General-Purpose Register file.
- Performance Monitor APU supporting execution profiling
- Nexus Class 3+ Real-time Development Unit
- Power management
 - Low power design with extensive clock gating
 - Power saving modes: DOZE, NAP, SLEEP, WAIT
 - Dynamic power management of execution units, caches, and local memories
- Testability
 - Synthesizeable, MuxD scan design
 - ABIST/MBIST for arrays
 - Built-in LBIST unit

13.3 Microarchitecture summary

The e200z420n3, e200z419, and e200z425n3 processor cores utilize a five stage instruction pipeline with two stages for execution. The Instruction Fetch, Instruction Decode/Register File Read/ EA Calc, Execute0/Memory Access0, Execute1/Memory Access1, and Register Writeback stages operate in an overlapped fashion allowing single-clock instruction execution for most instructions. [Figure 66](#) shows a block diagram of the e200z architecture.

The integer execution units each consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel Shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register Manipulation Unit (CRU), a Count Leading Zeros unit (CLZ), and result feed-forward hardware. Integer EU1 also supports hardware division and a 32 x 32 Hardware Multiplier array.

Most arithmetic and logical operations are executed in a single-cycle with the exception of the multiply, which is implemented with a pipelined hardware array, and divide instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

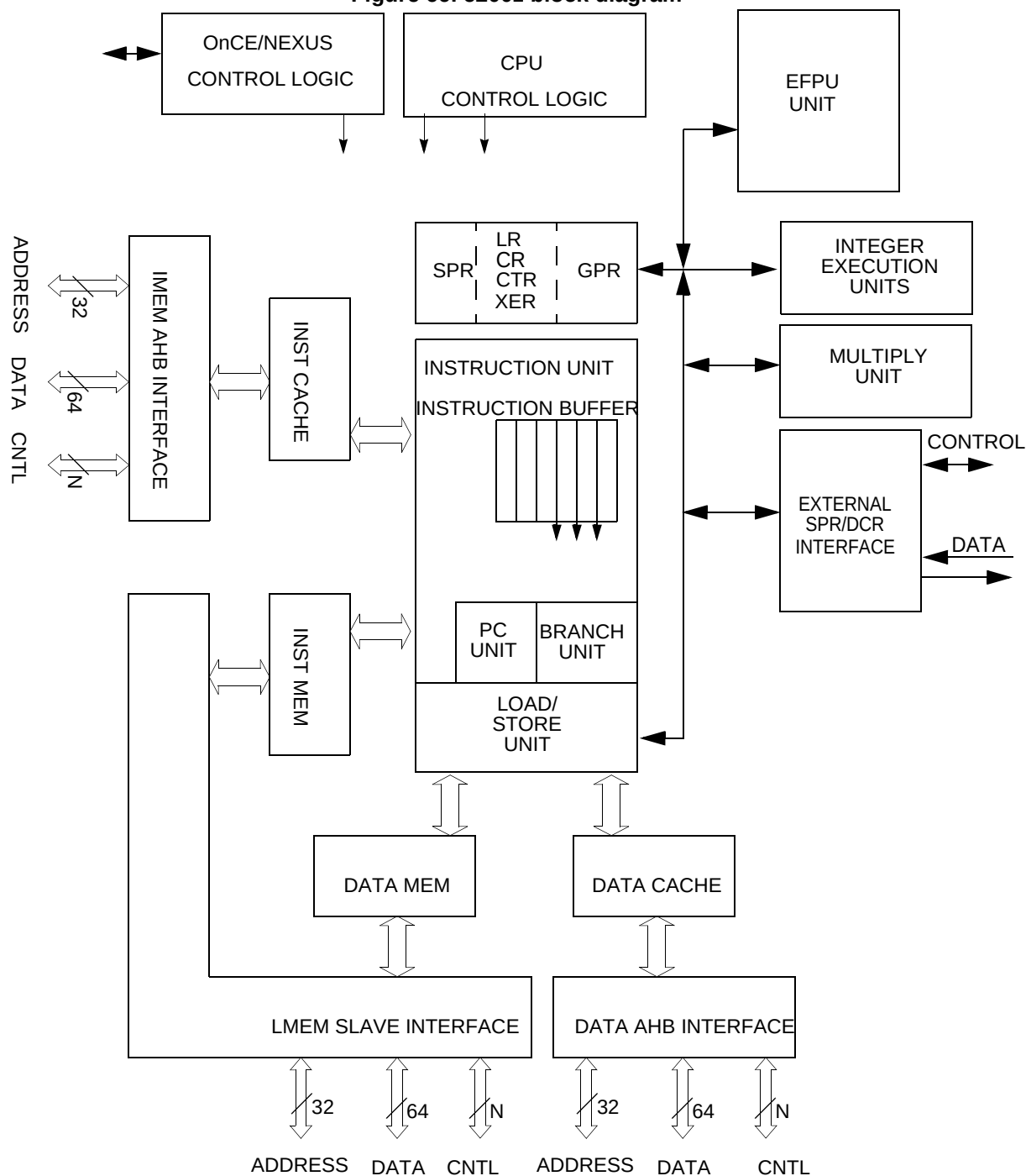
The Instruction Unit contains a PC incrementer and dedicated branch address adders to minimize delays during change-of-flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in an execution time of two clock cycles for correctly predicted branches. Conditional branches that are not taken execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, word (32-bit), and doubleword data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal. These instructions can be pipelined to allow effective single-cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to optimize effective address generation.

The CRU supports the condition register (CR) and condition register operations defined by the Power Architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations — such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Figure 66. e200z block diagram



Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The LSP APU (included only on the e200z425n3 core) supports vector instructions operating on 16-bit and 32-bit fixed-point data types. The EFPU2 APU supports 32-bit IEEE-754 single-precision floating-point formats and operates in a pipelined fashion. The 32-bit

general-purpose register file is used for source and destination operands. There is a unified storage model for scalar single-precision floating-point data types of 32 bits and the normal integer type. Low latency floating-point add, subtract, mixed add/subtract, sum, diff, min, max, multiply, multiply-add, multiply-sub, divide, square root, compare, and conversion operations are provided. Most operations can be pipelined.

13.3.1 Instruction unit features

The features of the e200z420n3, e200z419, and e200z425n3 instruction units are:

- 64-bit path to cache supports fetching of two 32-bit instructions per clock
- Instruction buffer holds up to 8 32-bit instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder and branch lookahead logic (BTB) supporting single-cycle execution of successfully predicted branches

13.3.2 Integer unit features

The e200z420n3, e200z419, and e200z425n3 integer units support single-cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for the Count Leading Zeros function
- 32-bit single-cycle Barrel Shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 4–14 clock cycles with minimized execution timing (EU1 only)
- Pipelined 32 x 32 hardware multiplier array supports 32 x 32->32 multiply with 2 clock latency, 1 clock throughput (EU1 only)

13.3.3 Load/Store unit features

The e200z420n3, e200z419, and e200z425n3 load/store units support load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions and context save/restore instructions

13.3.4 EFPU2 Floating-Point Unit features

The EFPU2 embedded floating-point unit supports pipelined operation of most floating-point operations, allowing a throughput of one FP operation per cycle, and supports a variety of operations:

- Supports IEEE754 single-precision data format
- Supports IEEE754 half-precision format for conversion $\leftrightarrow$ single-precision format
- EFPU2 instructions utilize the 32-bit GPRs for minimized context save/restore overhead
- Provides low latency pipelined floating-point add, subtract, add/sub, sub/add, multiply, multiply-add, min, max, absolute value, compare, and conversion instructions with single-cycle throughput
- Default results mode for real-time applications allows for exception-free operations when underflow or overflows occur
- Trap Enable controls allow for trapping of various operation exceptions and emulation of IEEE754 boundary case results
- Floating-point Divide and Square root logic operating in 13 (FP div) and 11 (FP sqrt) clock cycles

13.3.5 LSP Lightweight Signal Processing Unit features

The LSP APU is designed to accelerate signal processing applications normally suited to DSP operation.

- LSP instructions operate on the 32-bit GPRs
- Supports SIMD integer operations - arithmetic, logical, shift
- Supports SIMD rounding, saturation, pack, unpack, merge, extract, select for efficient data manipulation
- Supports SIMD multiply/multiply accumulate/ dot product operations
- 16, 32, and (limited) 64-bit signed and unsigned integer data types
- 16-bit and 32-bit signed fractional support (1.15, 1.31 representations)
- Guarded 32-bit and 64-bit signed fractional support (9.23, 33.31, and 17.47 representations)
- Single-cycle throughput for all instructions (except frac divide)
- Supports fractional divide operations (32/32 $\rightarrow$ 32)
- Specialized operations such as field manipulation, count-leading sign/zero bits
- Supports specialized load/store operations with integrated data manipulation
- Multiple addressing modes supported including circular and bit-reversed addressing
- Sustained throughput of up to 2 16x16-bit multiply/accumulate operations per cycle.

13.3.6 MPU features

The features of the MPU are as follows:

- 24-entry fully-associative Range Table
- Arbitrary range size support from 1 byte to 4 GB
- 8-bit process identifier
- Bypass capability for individual access types
- Maskable process identifier and address
- Programmable memory attributes
- Entry invalidation protection
- Write-once option
- Alternate Debug Breakpoint/Watchpoint function

13.3.7 Cache features

The features of the caches are as follows:

- 2-Way Set Associative Harvard Instruction and Data Caches
 - e200z420—8 KB ICache, 4 KB DCache
 - e200z425—8 KB ICache, No DCache
 - e200z419—No ICache, No DCache
- Writethrough support
- 8-entry Store Buffer
- Linefill Buffer
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and 64-bit write data bus
- Way locking support
- Write allocation policies
- Support for multi-bit EDC with correction/auto-invalidation capability for the instruction and data caches
- Hardware debug cache invalidate support for the data cache via DBL1CCSR0 register
- Software and hardware debug access to cache tag, status, and data storage via CDANTL and CDADATA dedicated control registers

13.3.8 Local memory features

The features of the Local Instruction and Data Memories are as follows:

- Local Instruction and Data Memories (IMEM, DMEM) with shared AHB 2.v6 64-bit slave interface
 - e200z420n3—16 KB IMEM, 64 KB DMEM
 - e200z425n3—16 KB IMEM, 64 KB DMEM
 - e200z419—No IMEM, No DMEM
- I-MEM provides RAM storage for instruction allowing the CPU to execute these instruction with 0 wait-state delay. I-MEM includes a feature to allow the memory to be relocated in the memory map as seen by the local CPU, such that it overlays MCU code flash (The memory map to other bus masters is unchanged). This relocation of the I-MEM into the flash region, allows it to be accessed directly from any location in

flash with a single short branch instruction. IMEM and DMEM are not duplicated on e200z419 core.

- Supports zero wait-state access relative to cache hit timings
- Supports multi-bit (SECCDED) ECC with correction/scrubbing capability
- 4-entry ECC RMW Store Buffer for DMEM
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and (for DMEM only) 64-bit write data bus

13.3.9 Performance Monitor Unit Features

The features of the e200z42x performance monitor include:

- Four configurable 32-bit performance monitor counters each capable of counting selected CPU subsystem events
- Performance Monitor interrupt
- Ability to configure performance monitor resources for debugger use
- Hardware input signals for qualification of counting by individual counters
- Hardware output signals to indicate counter overflows
- Dedicated watchpoint outputs for each counter with programmable periodicity
- Trigger On/Off control for each counter based on subsystem events
- Multiple selectable event types including # of clock cycles, CPU stall cycles, # of instructions completed, # of interrupts taken, # of memory accesses (by type), # of instruction or data cache misses, # of cycles with 0, 1, or 2 instructions issued, # of instructions in a given class completed (ldst, branch, integer, FP, etc.), cache linefills, and many other event types

13.3.10 e200z420n3 and e200z425n3 system bus features

The features of the e200z420n3 and e200z425n3 system bus interface are as follows:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture (AMBA) AHB2.v6 protocol
- 32-bit address bus, 64-bit data bus, plus attributes and control
- Data interface provides separate uni-directional 64-bit read and write data buses
- Support for HCLK running at a slower rate than CPU clock

13.3.11 e200z419 system features

Checker Core_0 is implemented using an e200z419 core. When enabled, Checker Core_0s is used in Lock Step with Main_Core_0. The e200z419 core can only be enabled or disabled by the user. For the SPC57EM80xx device, the Checker Core_0s has no system bus and cannot independently execute user code. The core is only used to execute the same instruction stream as Main Core 0.

The e200z419 core is identical to the e200z420n3 with five exceptions:

- IMEM memory array is not present
- I CACHE memory array is not present
- D MEM memory array is not present
- D CACHE memory array is not present
- The e200z419 has a pair of system bus interfaces, which are driven/sampled by the delayed Lock Step logic

13.4 Availability of detailed documentation

Detailed documentation of the e200z4xxd core is provided in a separate core reference manual (CRM). This CRM is available online at <http://www.st.com>.

14 Core e200z420n3 description

14.1 Overview of the e200z420n3 core

The e200z420n3 is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z420n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

The e200z420n3 processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

e200z420n3 contains an 8 KB Instruction Cache, and a 4 KB Data Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit that protects various instruction and data memory areas is also included.

14.2 Register model

This section describes the registers implemented in the e200z420n3 core. The *PowerISA 2.06* architecture defines register-to-register operations for all computational instructions.

e200z420n3 extends usage of the general purpose registers to support EFPU operations on the 32-bit GPR registers.

[Figure 67](#) and [Figure 68](#) show the complete e200z420n3 register set, all of which are accessible in supervisor mode. [Figure 69](#) and [Figure 70](#) show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

Figure 67. e200z420n3 Supervisor Mode Programmer's Model SPRs

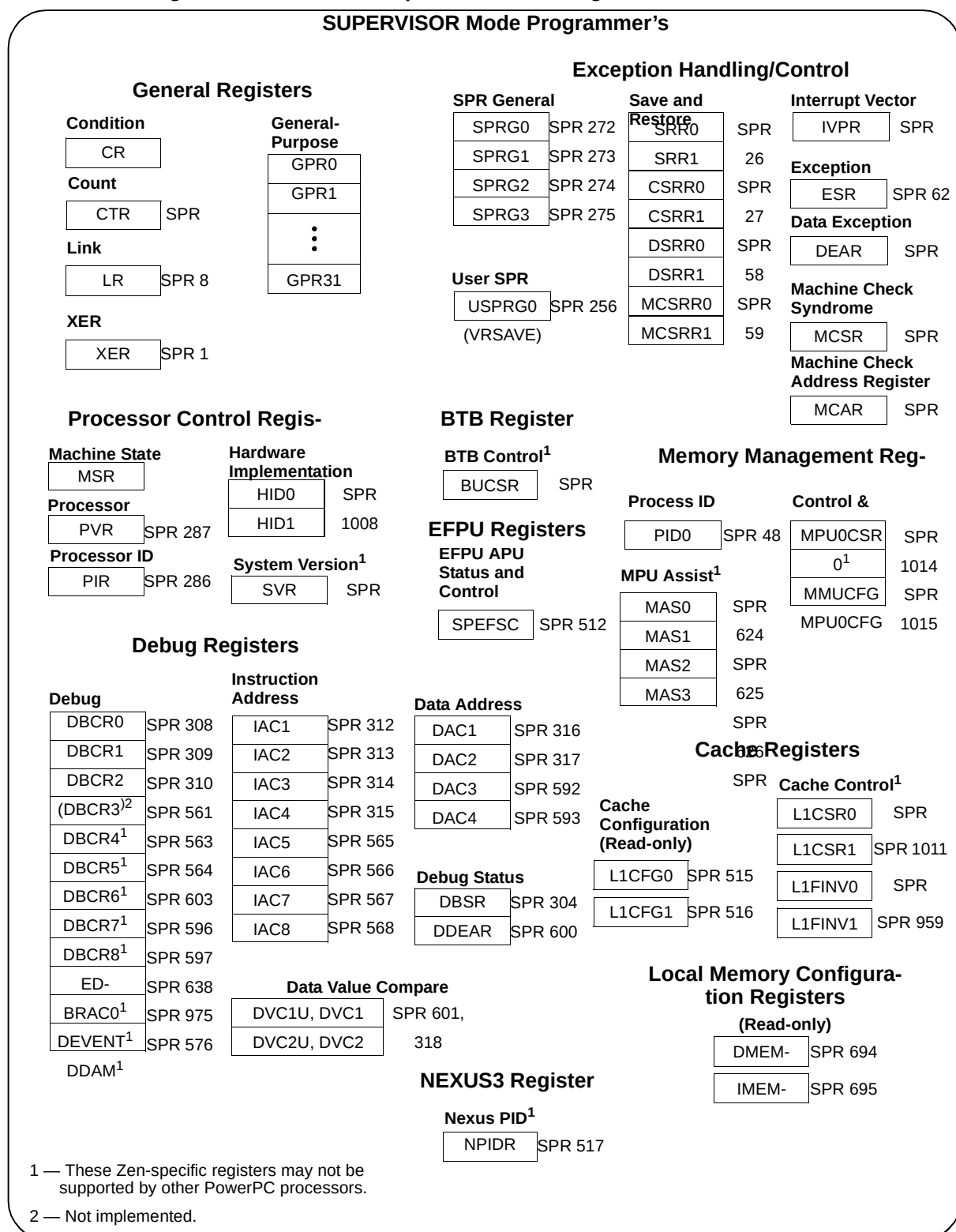


Figure 68. e200z420n3 Supervisor Mode Programmer's Model DCRs and PMRs

Supervisor Mode Programmer's Model DCRs

Performance Monitor Regis-

Control

PMGC0	PMR
PMLCa0	400
PMLCa1	PMR
PMLCa2	144
PMLCa3	PMR
PMLCb0	145
PMLCb1	PMR
PMLCb2	146
PMLCb3	PMR

User Control (read-only)

UPMGC0	PMR
UPMLCa0	384
UPMLCa1	PMR
UPMLCa2	128
UPMLCa3	PMR
UPMLCb0	129
UPMLCb1	PMR
UPMLCb2	130
UPMLCb3	PMR

Counters

PMC0	PMR 16
PMC1	PMR 17
PMC2	PMR 18
PMC3	PMR 19

User Counters

UPMC0	PMR 0
UPMC1	PMR 1
UPMC2	PMR 2
UPMC3	PMR 3

Nexus3 Debug Reg-

DC1	DCR
DC2	368
DC3	DCR
DC4	369
WT	DCR
DTC	370
DTSA1	DCR
DTSA2	371
DTSA3	DCR
DTSA4	375
DTEA1	DCR
DTEA2	376
DTEA3	DCR
DTEA4	377
DS	DCR
OVCR	378
WMSK	DCR
PTSTC	379
PTETC	DCR
DTSTC	380
DTETC	DCR
	381
	DCR
	382
	DCR

Cache Access Reg-

CDACNT
L

Local Memory Reg-

DMEMCT	DCR
L0	496
IMEMCTL	DCR
0	497

e2eECC Registers¹

E2ECTL0	DCR
E2EECSR	510

1 — These Zen-specific registers may not be supported by other PowerISA processors.

Figure 69. e200z420n3 User Mode Programmer's Model SPRs

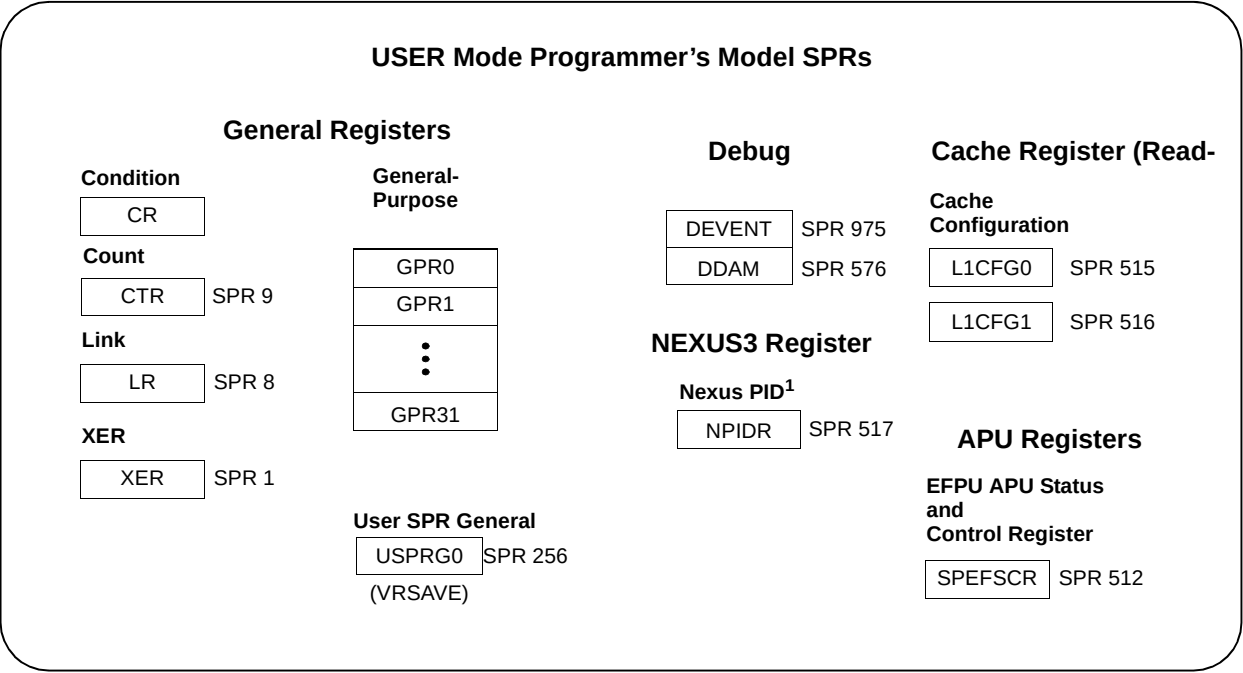
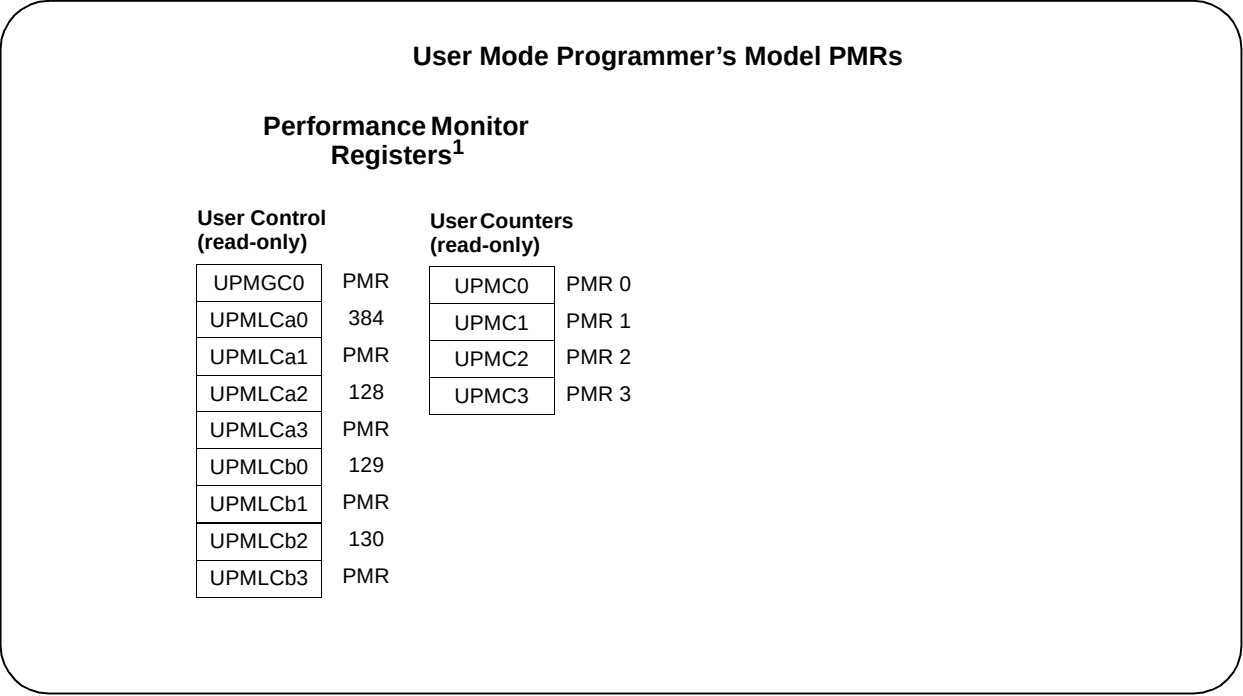


Figure 70. e200z420n3 User Mode Programmer's Model PMRs



14.3 Dual-issue operation

The instruction issue unit attempts to issue a pair of instructions to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the

operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions.

Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

[Table 143](#) shows the e200z420n3 concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue of those instructions.

Table 143. Concurrent instruction issue capabilities for e200z420n3

Class of instruction	Branch	Load/Store	Scalar integer	Scalar float	Special
Branch	—	yes	yes	yes	—
Load/Store	yes	—	yes	yes	—
Scalar integer	yes	yes	yes ⁽¹⁾	yes	—
Scalar float	yes	yes	yes	—	—
Special	—	—	—	—	—

1. Excludes multiply or divide class instructions occurring in both issue slots.

14.4 Reservation instructions and cache interactions

The CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

14.5 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in Figure 71.

Figure 71. EFPU Status and Control Register (SPEFSCR)

0										FINXS	FINVS	FDBZS	FUNFS	FOVFS	MODE	0	0	FG	FX	FINV	FDBZ	FUNF	FOVF	0	FINXE	FINVE	FDBZE	FUNFE	FOVFE	FRMC	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 512; Read/Write; Reset: 0x0

The SPEFSCR bits are defined in [Table 144](#).

Table 144. EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z420n3 supports only mode 0. Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by Zen)
16 (48)	—	Reserved
17 (49)	—	Reserved
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler. FX is zeroed if a Floating-point Data Exception occurs.

Table 144. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the I dividend is a finite nonzero number.
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact ($FG \vee FX = 1$), or if the result of a Floating-point instruction does result in overflow ($FOVF = 1$) but Floating-point Overflow exceptions are disabled ($FOVFE = 0$), or if the result of a Floating-point instruction results in underflow ($FUNF = 1$ or $FUNFH = 1$) but Floating-point Underflow exceptions are disabled ($FUNFE = 0$), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled

Table 144. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 (62:63)	FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

14.6 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

14.6.1 Cache overview

The e200z420n3 processor supports an 8 KB 2-way set-associative instruction and 4 KB 2-way set-associative data cache with a 32-byte line size.

The caches improve system performance by providing low-latency data to the e200z420n3 instruction and data pipelines, which decouples processor performance from system memory performance.

The e200z420n3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction and data addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

14.6.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache and for disabling ways in both caches. The L1CSR0 register is accessed using a **mfsprr** or **mtsprr** instruction. The SPR number for L1CSR0 is 1010 in decimal. The L1CSR0 register is shown in [Figure 72](#).

Figure 72. L1 Cache Control and Status Register 0 (L1CSR0)

WID		0		WDD		0						DCWA	0		DCECE	DCEI	DCLOC		0						DCEA		DCLOINV	0		DCABT	DCINV	DCE
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	

SPR 1010; Read/Write; Reset: 0x0

The L1CSR0 fields are described in [Table 145](#).

Table 145. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:3	—	Reserved ⁽¹⁾
4:5	WDD	Way Data Disable Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache. 0 The corresponding way in the data cache is available for replacement by data miss line fills. 1 The corresponding way in the data cache is not available for replacement by data miss line fills.
8:11	—	Reserved ⁽¹⁾
12	DCWA	Data Cache Write Allocation Policy This bit also controls merging of store data into the linefill buffer while a cache linefill is in progress. Store data will not be merged when write allocation is disabled. 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled
13:14	—	Reserved ⁽¹⁾
15	DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	DCEI	Data Cache Error Injection DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits (p_dchk[0:1]). This includes writes due to store hits as well as writes due to cache line refills.

Table 145. L1CSR0 field descriptions (continued)

Bits	Name	Description
17:18	DCLOC	Data Cache Lockout Control <i>Note: For the dcbi and dcbf instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled.</i> <i>Note: When operating in MC mode (DCEA = 00), detected errors on cache-inhibited accesses will not cause a data cache line to be locked out, instead the cache line contents are ignored.</i> 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved ⁽¹⁾
25:26	DCEA	Data Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	DCLOINV	Data Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing DCINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved ⁽¹⁾
29	DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

Table 145. L1CSR0 field descriptions (continued)

Bits	Name	Description
30	DCINV	Data Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. 0 No cache invalidate 1 Cache invalidation operation
31	DCE	Data Cache Enable When disabled, cache lookups are not performed for normal load or store accesses. Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

14.6.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfsprr** or **mtsprr** instruction. The SPR number for L1CSR1 is 1011 in decimal. The L1CSR1 register is shown in [Figure 73](#).

Figure 73. L1 Cache Control and Status Register 1 (L1CSR1)

0															ICECE	ICEI	ICLOC	0							ICEA	ICLOINV	0	ICABT	ICINV	ICE	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 1011; Read/Write; Reset: 0x0

The L1CSR1 fields are described in [Table 146](#).

Table 146. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved ⁽¹⁾
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits (p_chk[0:1]).

Table 146. L1CSR1 field descriptions (continued)

Bits	Name	Description
17:18	ICLOC	Instruction Cache Lockout Control <i>Note: For the icbi instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled.</i> <i>Note: When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored.</i> 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved ⁽¹⁾
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved ⁽¹⁾
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

Table 146. L1CSR1 field descriptions (continued)

Bits	Name	Description
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

14.6.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z420n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspir** instruction. The SPR number for L1CFG0 is 515 in decimal. The L1CFG0 register is shown in [Figure 74](#).

Figure 74. L1 Cache Configuration Register 0 (L1CFG0)

CARCH	CWPA	CFAHA	DCFISWA	0	DCBSIZE	DCREPL	DCLA	DCECA	DCNWAY												DCSIZE											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0	0	1	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0

SPR 515; Read-only

The L1CFG0 bits are described in [Table 147](#).

Table 147. L1CFG0 field descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available 1 The caches support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 The data cache does not support Flush All in Hardware

Table 147. L1CFG0 field descriptions (continued)

Bits	Name	Description
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 The data cache supports invalidation by Set and Way via L1FINV0
5:6	—	Reserved — read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11 The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x004 The size of the data cache is 4 KB

14.6.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z420n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspir** instruction. The SPR number for L1CFG1 is 516 in decimal. The L1CFG1 register is shown in [Figure 75](#).

Figure 75. L1 Cache Configuration Register 1 (L1CFG1)

0	ICFISWA	0	ICBSIZE	ICREPL	ICLA	ICECA	ICNWAY	ICSIZE
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0

SPR 516; Read-only

The L1CFG1 fields are described in [Table 148](#).

Table 148. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved — read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved — read as zeros

Table 148. L1CFG1 field descriptions (continued)

Bits	Name	Description
7:8	ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 The size of the instruction cache is 8 KB

14.6.6 Data Cache Software Coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

14.6.7 Data Cache Hardware Coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency.

Debug support is provided, however, for a D-Cache line invalidation operation requested by an external hardware debugger. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available D-Cache access cycle

14.6.8 Cache Invalidate by Set and Way

e200z420n3 supports cache set/way invalidation under software control. The caches may be invalidated by index and way through a **mtspr l1finv{0,1}** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0,1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

14.6.8.1 L1FINV0

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in [Figure 76](#).

Figure 76. L1 Flush/Invalidate Register 0 (L1FINV0)

0	CWAY	0	CSET	0	CCMD
---	------	---	------	---	------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 1016; Read/Write; Reset: 0x0

The L1FINV0 bits are described in [Table 149](#).

Table 149. L1FINV0 field descriptions

Bits	Name	Description
0:6	—	Reserved ⁽¹⁾ for way extension
7	CWAY	Cache Way Specifies the data cache way to be selected
8:20	—	Reserved ⁽¹⁾ for set extension
21:26	CSET	Cache Set Specifies the cache set to be selected
27:28	—	Reserved ⁽¹⁾ for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

14.6.8.2 L1FINV1

The SPR number for L1FINV1 is 959 in decimal. The L1FINV1 register is shown in [Figure 77](#).

Figure 77. L1 Flush/Invalidate Register 1 (L1FINV1)

0	CWAY	0	CSET	0	CCMD
---	------	---	------	---	------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 959; Read/Write; Reset: 0x0

The L1FINV1 bits are described in [Table 150](#).

Table 150. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved ⁽¹⁾ for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8:19	—	Reserved ⁽¹⁾ for set extension
20:26	CSET	Cache Set Specifies the instruction cache set to be selected
27:28	—	Reserved ⁽¹⁾ for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

14.6.9 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of each cache. Seven parity check bits are provided for each tag entry for the tag arrays of both caches to support error detection and correction (ECC — SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, and each word in the data arrays of the D-Cache, which are used for single- and double-bit error detection (EDC — DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the L1CSR0_{DCECE} and L1CSR1_{ICECE} control fields. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the respective cache when cache error checking is disabled for that cache (L1CSR{0,1}_{[D,I]CECE} = 0).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, normal loads, or normal stores, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Store hits to the D-Cache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to

the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full-width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the D-Cache during otherwise idle cycles when two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

14.6.9.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the I-Cache and D-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by $L1CSR\{0,1\}_{[D,I]CLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on normal instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via $L1CSR\{0,1\}_{[D,I]CLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch or normal load or store access, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, to avoid certain exception conditions and for consistency with error reporting, specialized load/store accesses do not set LO bits; instead, cache contents are ignored for lines with those errors. Cache flush and invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable normal access will perform the lockout function if required.

14.6.10 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If L1CSR1_{ICEI} is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If L1CSR0_{DCEI} is set, any cache line fill to the data cache data array has the associated two most significant parity check bits inverted in the data array for each word loaded. Additionally, inverted parity bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction, or an L1FINV{0,1} invalidation operation.

14.7 Exceptions

Interrupts implemented in e200z420n3 and the exception conditions that cause them are listed in [Table 151](#).

Table 151. Exceptions and conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] 2'b00	Reset
Critical Input	0x00 ⁽¹⁾	p_critint_b is asserted and MSR _{CE} = 1.
Machine check	0x10	1. p_mcp_b transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. Parity Error signaled on Local Memory access 5. External bus error 6. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40 ⁽²⁾	Interrupt Controller interrupt and MSR _{EE} = 1
Alignment	0x50	1. lmw, stmw not word aligned 2. lwarx or stwcx. not word aligned, lharx or sthcx. not halfword aligned 3. dcbz
Program	0x60	Illegal, Privileged, Trap

Table 151. Exceptions and conditions (continued)

Interrupt type	Interrupt vector offset value	Causing conditions
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/PMGC0 _{UDI} = 0
System call	0x80	Execution of the System Call (se_sc) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/PMGC0 _{UDI} = 1, MPU
EFPU Data Exception	0xA0	See section “Embedded Floating-point Data Exception” in the Power Architecture core reference manual relative to this core.
EFPU Round Exception	0xB0	See section “Embedded Floating-point Round Exception” in the Power Architecture core reference manual relative to this core.
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

14.7.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.

Figure 78. Exception Syndrome Register (ESR)

0				PIL	PPR	PTR	FP	ST	0		0	0	BO	0	0								SPV	0	VLEMI	0			0	0	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 62; Read/Write; Reset: 0x0

The ESR bits are defined in [Table 152](#).

Table 152. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For core e200z420n3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage

Table 152. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

14.7.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z420n3 MSR is shown in [Figure 79](#).

Figure 79. Machine State Register (MSR)

0														CE	0	EE	PR	0	ME	0	0	DE	0							PMM	RI	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	

Read/ Write; Reset: 0x0

The MSR bits are defined in [Table 153](#).

Table 153. MSR field descriptions

Bit(s)	Name	Description
0:13	—	Reserved
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	—	Reserved
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	—	Reserved
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23:28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR _{PR} and MSR _{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLC _n Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

14.7.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in [Figure 80](#).

Figure 80. Machine Check Syndrome Register (MCSR)

0	MCP	1	IC_DPERR	2	0	3	DC_DPERR	4	EXCP_ERR	5	IC_TPERR	6	DC_TPERR	7	IC_LKERR	8	DC_LKERR	9	0	10		11	NMI	12	MAV	13	MEA	14	U	15	IF	16	LD	17	ST	18	G	19	0	20		21	STACK_ERR	22	IMEM_PERR	23	DMEM_RDPERR	24	DMEM_WRPERR	25	0	26		27	BUS_IRERR	28	BUS_DRERR	29	BUS_WRERR	30	BUS_WRDSI	31	0
---	-----	---	----------	---	---	---	----------	---	----------	---	----------	---	----------	---	----------	---	----------	---	---	----	--	----	-----	----	-----	----	-----	----	---	----	----	----	----	----	----	----	---	----	---	----	--	----	-----------	----	-----------	----	-------------	----	-------------	----	---	----	--	----	-----------	----	-----------	----	-----------	----	-----------	----	---

SPR 572; Read/Clear; Reset: 0x0

[Table 154](#) describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as “write 1 to clear.” Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once MSR_{ME} = 1.

Table 154. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type ⁽¹⁾	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—

Table 154. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type ⁽¹⁾	Recoverable
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1, MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address ⁽²⁾	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise

Table 154. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type ⁽¹⁾	Recoverable
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.

— “Error Report” indicates that this bit is only set for error report exceptions that cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

— “Status” indicates that this bit provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

— “NMI” indicates that this bit is only set for the non-maskable interrupt type exception, which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a ‘1’ to the bit position.

— “Async Mchk” indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any “Async Mchk” bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME} = 1. If MSR_{ME} = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

2. Note that since no address translation mechanism is present on e200z420n3, the MEA value is always set to ‘0’.

14.7.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in [Figure 81](#).

Figure 81. Zen Interrupt Vector Prefix Register (IVPR)

Vector Base																								0							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 63; Read/Write

The IVPR fields are defined in [Table 155](#).

Table 155. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

14.7.5 Interrupt Definitions

14.7.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE} , the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

[Table 156](#) lists register settings when a Critical Input interrupt is taken.

Table 156. Critical Input Interrupt—register settings

Register	Setting description		
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
CSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME — FE0 0 DE —/0 ⁽¹⁾	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	IVPR _{0:23} 0x00 (autovectored) IVPR _{0:15} (IVPR _{16:29} p_voffset[0:13]) 2'b00 (non-autovectored)		

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

14.7.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register (HID0_{MCCLRDE}).

When a Machine Check interrupt is taken, registers are updated as shown in [Table 157](#).

Table 157. Machine Check Interrupt—register settings

Register	Setting description		
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.		
MCSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0/— ⁽¹⁾	FE1 0 IS 0 DS 0 PMM 0 RI 0
ESR	Unchanged		
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.		
Vector	IVPR _{0:23} 0x10		

1. Clearing of DE is optionally supported by control in HID0.

14.7.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

[Table 158](#) lists register settings when a DSI is taken.

Table 158. Data Storage Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting load/store instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Access:	[ST], [SPV], VLEMI. All other bits cleared.	
MCSR	Unchanged		

Table 158. Data Storage Interrupt—register settings (continued)

Register	Setting description
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.
Vector	IVPR _{0:23} 0x20

14.7.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

[Table 159](#) lists register settings when an ISI is taken.

Table 159. Instruction Storage Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	IVPR _{0:23} 0x30		

14.7.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

[Table 160](#) lists register settings when an External Input interrupt is taken.

Table 160. External Input Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table 160. External Input Interrupt—register settings (continued)

Register	Setting description			
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —	
ESR	Unchanged			
MCSR	Unchanged			
DEAR	Unchanged			
Vector	IVPR _{0:23} 0x40 (autovectored) IVPR _{0:15} (IVPR _{16:29} p_voffset[0:13]) 2'b00 (non-autovectored)			

14.7.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

[Table 161](#) lists register settings when an alignment interrupt is taken.

Table 161. Alignment Interrupt—register settings

Register	Setting description			
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.			
SRR1	Set to the contents of the MSR at the time of the interrupt			
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —	
ESR	[ST], VLEMI. All other bits cleared.			
MCSR	Unchanged			
DEAR	Set to the effective address of a byte of the load or store access causing the violation.			
Vector	IVPR _{0:23} 0x50			

14.7.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z420n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspir** instructions with an undefined SPR specified
- **mtdcr** and **mfddcr** instructions with an undefined DCR specified

The e200z420n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when $MSR_{PR} = 1$ (user mode):

- A privileged instruction
- **mtspr** and **mfspir** instructions that specify a SPRN value with $SPRN_5 = 1$ (even if the SPR is undefined)

The e200z420n3 will invoke a Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfddcrx**, **mtddcrx**, or on any *PowerISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

[Table 162](#) lists register settings when a Program interrupt is taken.

Table 162. Program Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Illegal, Unimplemented: Privileged: Trap:	PIL, VLEMI. All other bits cleared. PPR, VLEMI. All other bits cleared. PTR, VLEMI. All other bits cleared.	
MCSR	Unchanged		
DEAR	Unchanged		
Vector	IVPR _{0:23} 0x60		

14.7.5.8 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

[Table 163](#) lists register settings when a System Call interrupt is taken.

Table 163. System Call Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the instruction <i>following</i> the system call instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	IVPR _{0:23} 0x80		

14.7.5.9 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both DBCR0_{IDM} = 1 (internal debug mode) and MSR_{DE} = 1).

[Table 164](#) lists register settings when a Debug interrupt is taken.

Table 164. Debug Interrupt—register settings

Register	Setting description		
DSRR0 ⁽¹⁾	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP. Set to the effective address of the next instruction to be executed <i>following</i> the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
DSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE —/0 ⁽²⁾ EE —/0 ⁽²⁾ PR 0	FP 0 ME — FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI —

Table 164. Debug Interrupt—register settings (continued)

Register	Setting description	
DBSR ⁽³⁾	Unconditional Debug Event:	UDE
	Instr. Complete Debug Event:	ICMP
	Branch Taken Debug Event:	BRT
	Interrupt Taken Debug Event:	IRPT
	Critical Interrupt Taken Debug Event:	CIRPT
	Trap Instruction Debug Event:	TRAP
	Instruction Address Compare:	{IAC}
	Data Address Compare:	{DACR, DACW}
	Debug Notify Interrupt:	DNI
	Return Debug Event:	RET
	Critical Return Debug Event:	CRET
	External Debug Event:	{DEVT1, DEVT2}
	Performance Monitor Debug Event:	PMI
	MPU Debug Event:	MPU
	and optionally, an Imprecise Debug Event flag	{IDE}
ESR	Unchanged	
MCSR	Unchanged	
DEAR	Unchanged	
Vector	IVPR _{0:23} 0x90	

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

14.7.5.10 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

[Table 165](#) lists register settings when a System Reset interrupt is taken.

Table 165. System Reset Interrupt—register settings

Register	Setting description
CSRR0	Undefined
CSRR1	Undefined

Table 165. System Reset Interrupt—register settings (continued)

Register	Setting description			
MSR	SPV 0	FP 0	FE1 0	
	WE 0	ME 0	IS 0	
	CE 0	FE0 0	DS 0	
	EE 0	DE 0	PMM 0	
	PR 0		RI 0	
ESR	Cleared			
DEAR	Undefined			
Vector	[p_rstbase[0:29]] 2'b00			

14.7.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

[Table 166](#) lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 166. Embedded Floating-point Data Interrupt—register settings

Register	Setting description			
SRR0	Set to the effective address of the excepting EFPU instruction.			
SRR1	Set to the contents of the MSR at the time of the interrupt			
MSR	SPV 0	FP 0	FE1 0	
	WE 0	ME —	IS 0	
	CE —	FE0 0	DS 0	
	EE 0	DE —	PMM 0	
	PR 0		RI —	
ESR	SPV, VLEMI. All other bits cleared.			
MCSR	Unchanged			
DEAR	Unchanged			
Vector	IVPR _{0:23} 0xA0			

14.7.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

[Table 167](#) lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 167. Embedded Floating-point Round Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table 167. Embedded Floating-point Round Interrupt—register settings (continued)

Register	Setting description			
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —	
ESR	SPV, VLEMI. All other bits cleared.			
MCSR	Unchanged			
DEAR	Unchanged			
Vector	IVPR _{0:23} 0xB0			

14.7.5.13 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMCx register overflow condition occurs with the following settings:

- PMLCax_{CE} = 11; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, PMGC0_{PMIE} must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

[Table 168](#) lists register settings when an performance monitor interrupt is taken.

Table 168. Performance Monitor Interrupt—Register Settings

Register	Setting description			
SRR0/ DSRR0 ⁽¹⁾	Set to the effective address of the next instruction to be executed.			
SRR1/ DSRR1 ⁽¹⁾	Set to the contents of the MSR at the time of the interrupt.			
MSR	SPV 0 WE 0 CE —/0 ⁽²⁾ EE 0/— ⁽³⁾ PR 0	FP 0 ME — FE0 0 DE —/0 ⁽⁴⁾	FE1 0 IS 0 DS 0 PMM 0 RI —	
ESR	Unchanged			
MCSR	Unchanged			
DEAR	Unchanged			
Vector	IVPR _{0:23} 0x70			

1. DSRR0/1 are used if PMGC0_{UDI} = 1

2. CE is cleared if PMGC0_{UDI} = 1 and HID0_{DCLRC} = 1

3. EE is not cleared if $PMGC0_{UDI} = 1$ and $HID0_{DCLREE} = 0$
4. DE is cleared if $PMGC0_{UDI} = 1$

14.8 Memory Protection Unit (MPU)

14.8.1 MPU Overview

The e200z420n3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

14.8.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mf spr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

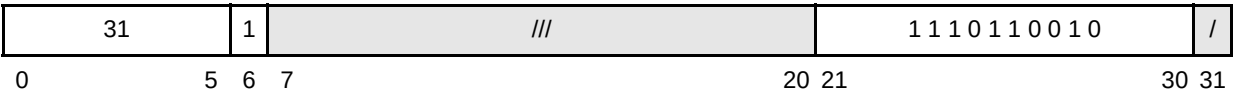
14.8.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure

mpu read entry

mpure



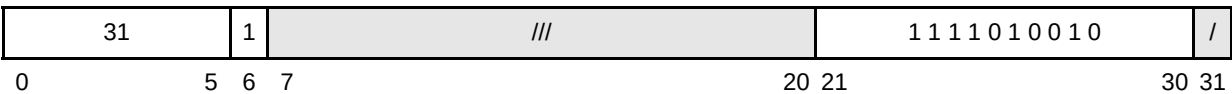
```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

14.8.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe
mpu write entry

mpuwe



```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

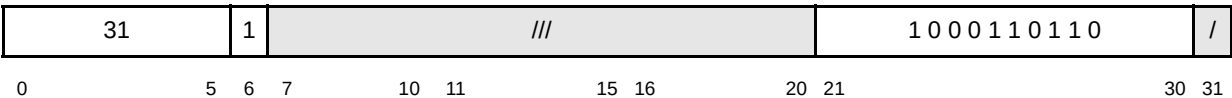
14.8.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the e200z420n3.

mpusync
MPU Synchronize

mpusync

mpusync



14.8.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z420n3 MPU design. The MMUCFG register is shown in [Figure 82](#).

Figure 82. MMU/MPU Configuration Register (MMUCFG)

0	RASIZE	0	PIDSIZE	NMPUS	NTLBS	MAVN
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31						

SPR 1015; Read-Only

The MMUCFG bits are described in [Table 169](#).

Table 169. MMUCFG field descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

14.8.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z420n3 MPU. The MPU0CFG register is shown in [Figure 83](#).

Figure 83. MPU0 Configuration Register (MPU0CFG)

FASSOC	0	MINSIZE	MAXSIZE	IPROT	0	UAMSKA	0	SHENTRY	DENTRY	IENTRY
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31										

SPR 692; Read-Only

The MPU0CFG bits are described in [Table 170](#).

Table 170. MPU0CFG field descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15	MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16	IProt	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28	DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

14.8.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in [Figure 84](#).

Figure 84. MPU0 Control and Status Register 0 (MPU0CSR0)

0																BYPSR	BYPSW	BYPSX	BYPUR	BYPUW	BYPUX	0	DRDEN	DWDEN	IDEN	0	TIDCTL	0	MPUFI	MPUIEN	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 1014; Read/ Write; Reset: 0x0⁽¹⁾

- Reset by processor reset **p_reset_b** if EDBCR0_{EDM} = 0, as well as unconditionally by **m_por**. If EDBCR0_{EDM} = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Specifically, this applies to the DRDEN, DWDEN, and IDEN control bits.

The MPU0CSR0 bits are described in [Table 171](#).

Table 171. MPU0CSR0 field descriptions

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17	BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19	BYPUR	Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses
20	BYP UW	Bypass MPU protections for User Write accesses This bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21	BYP UX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
22:23	—	Reserved

Table 171. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
24	DRDEN	Data Read Debug Enable This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses
25	DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses 1 MPU debug events are enabled for data write accesses
26	IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses 1 MPU debug events are enabled for instruction accesses
27	—	Reserved
28	TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode
29	—	Reserved

Table 171. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
30	MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. 0 No flash invalidate 1 MPU1 invalidation operation <i>Note: Entries that have the IPROT bit set will not be invalidated.</i>
31	MPUEN	MPU Enable This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

14.8.9 MPU Assist Registers (MAS)

The e200z420n3 uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mf spr** and **mt spr** instructions.

The MAS0 register is shown in [Figure 85](#). Fields are defined in [Table 172](#).

Figure 85. MPU Assist Register 0 (MAS0)

VALID	IPROT	SEL		0	RO	DEBUG	INST	SHD	0		ESEL				0	UAMSK				UW	SW	UX/UR	SX/SR	IOVR	GOVR ⁽¹⁾	1	1	I	0	G ⁽¹⁾	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 624; Read/ Write; Reset: Unaffected

1. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Table 172. MAS0—MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 This MPU entry is invalid 1 This MPU entry is valid
1	IPROT	Invalidation Protect 0 Entry is not protected from invalidation 1 Entry is protected from invalidation

Table 172. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
2:3	SEL	Selects MPU for access: 00 Reserved, no access 01 Reserved, no access 10 Select MPU 11 Reserved, no access
4	—	Reserved
5	RO	Read-Only When set to 1, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs. 0 This MPU entry is writable 1 This MPU entry is Read-only
6	DEBUG	Debug Control for Entry This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access. 0 This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access
7	INST	Instruction Entry When SHD = 0, this bit is used to select the INST entry portion of the region descriptor table for mpure and mpuwe instruction operations. When SHD = 1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. 0 This MPU entry is used for matching data accesses only 1 This MPU entry is used for matching instruction accesses only
8	SHD	Shared Entry Select This bit is used to control selection of the Shared portion of the region descriptor table. 0 The shared portion of the region descriptor table is not accessed on a mpure or mpuwe operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 The shared portion of the region descriptor table is accessed on a mpure or mpuwe operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected.
9:11	—	Reserved
12:15	ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved

Table 172. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
17:19	UAMSK	Upper Address Mask Control This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly. 000 No upper address bits are masked, address matching uses all 32 address bits for accesses 001 The most significant access address bit is forced to 0 before matching 010 The two most significant access address bits are forced to 0 before matching 011 The three most significant access address bits are forced to 0 before matching 100 The four most significant access address bits are forced to 0 before matching 101 The upper five access address bits are forced to 0 before matching 11x Reserved, do not use
20	UW	User Mode Write Permission Determines User mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 No User mode Write permission 1 User mode has Write permission
21	SW	Supervisor Mode Write / Read Permission Determines Supervisor mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 No Supervisor mode Write permission 1 Supervisor mode has Write permission
22	UX/UR	User Mode Execute / Read Permission Determines User mode Execute (X) permission when INST = 1, or User mode Read (R) permission when INST = 0. 0 No User mode Execute/ Read permission 1 User mode has Execute/ Read permission
23	SX/SR	Supervisor Mode Execute / Read Permission Determines Supervisor mode Execute (X) permission when INST = 1, or Supervisor mode Read (R) permission when INST = 0. 0 No Supervisor mode Execute/ Read permission 1 Supervisor mode has Execute/ Read permission
24	IOVR	Cache-Inhibit attribute Override Determines whether this matching entry overrides the I bit settings of other matching entries and the cache-inhibited region configuration signals. 0 No override of I attribute by entry 1 Entry I bit overrides I attribute
25	GOVR/—	G attribute Override Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 No override of G attribute by entry 1 Entry G bit overrides G attribute
26	—	Reserved
27	—	Reserved

Table 172. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
28	I	Cache Inhibited This attribute may be overridden by the cache-inhibited region configuration input settings. 0 This region is considered cacheable 1 This region is considered cache-inhibited
29	—	Reserved
30	G/—	Guarded This attribute may be overridden by the guarded region configuration input settings. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 All loads and stores to this region are performed without speculation (i.e. they are known to be required)
31	—	Reserved

The MAS1 register is shown in [Figure 86](#). Fields are defined in [Table 173](#).

Figure 86. MPU Assist Register 1 (MAS1)

0	TID	0	TIDMSK
---	-----	---	--------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 625; Read/Write; Reset: Unaffected

Table 173. MAS1—descriptor context and configuration control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.

Table 173. MAS1—descriptor context and configuration control

Bit	Name	Comments, or Function when Set
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx TID[0] comparison to PID0[0] not masked 1xxxxxxx TID[0] comparison to PID0[0] is masked x0xxxxxx TID[1] comparison to PID0[1] not masked x1xxxxxx TID[1] comparison to PID0[1] is masked xx0xxxxx TID[2] comparison to PID0[2] not masked xx1xxxxx TID[2] comparison to PID0[2] is masked xxx0xxxx TID[3] comparison to PID0[3] not masked xxx1xxxx TID[3] comparison to PID0[3] is masked xxxx0xxx TID[4] comparison to PID0[4] not masked xxxx1xxx TID[4] comparison to PID0[4] is masked xxxxx0xx TID[5] comparison to PID0[5] not masked xxxxx1xx TID[5] comparison to PID0[5] is masked xxxxxx0x TID[6] comparison to PID0[6] not masked xxxxxx1x TID[6] comparison to PID0[6] is masked xxxxxxx0 TID[7] comparison to PID0[7] not masked xxxxxxx1 TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in [Figure 87](#). Fields are defined in [Table 174](#).

Figure 87. MPU Assist Register 2 (MAS2)

UPPER BOUND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SPR 626; Read/ Write; Reset: Unaffected																															

Table 174. MAS2—Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND ≤ EA ≤ UPPER_BOUND

The MAS3 register is shown in [Figure 88](#). Fields are defined in [Table 175](#).

Figure 88. MPU Assist Register 3 (MAS3)

LOWER BOUND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SPR 627: Read/ Write: Reset: Unaffected																															

Table 175. MAS3—Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND ≤ UPPER_BOUND

14.8.10 MAS Registers Summary

The MAS registers are summarized in [Figure 89](#).

Figure 89. MPU Assist Registers Summary

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MAS0	VALID	IPROT	SEL		0	RO	DEBUG	INST	SHD	0		ESEL				0	UAMSK				UW	SW	UX/JR	SX/SR	IOVR	GOVR ⁽¹⁾	1	1	I	0	G ⁽¹⁾	0
MAS1	0								TID								0								TIDMSK							
MAS2	UPPER BOUND																															
MAS3	LOWER BOUND																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

1. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1.

14.9 Local memories

14.9.1 Local Instruction and Data memory overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added. This provides low latency access (zero wait-states) similar to an instruction or data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large capacity “tightly coupled” storage capability.

14.9.2 Local memory control and configuration

DMEM and IMEM local memory configuration information is provided by a set of privileged read-only SPRs that are accessed using **mf spr** instructions. DMEM and IMEM local memory operation is controlled by a set of privileged device control registers (DCRs) that are accessed using the **mf dcr** and **mt dcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

14.9.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z420n3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mf spr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register

is shown in [Figure 90](#).

Figure 90. DMEM Configuration Register 0 (DMEMCFG0)

DMEM BASE ADDR																				0	DECUA	DECA	0	DMSIZE					0		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CURRENT DMEM BASE ADDRESS VALUE																				0	1	1	0	010000 (64 KB)					00		
SPR 694; Read-only; Privileged																															

The DMEMCFG0 fields are described in [Table 176](#).

Table 176. DMEMCFG0 field descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. <i>Note: Low order bits of this field are driven to 0s when the DMEM size is > 4 KB</i>
20	—	Reserved
21	DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available. 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access.
22	DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available. 1 Error Correction is available.
23	—	Reserved
24:29	DMSIZE	DMEM Size 010000 The size of the DMEM is 64 KB.
30:31	—	Reserved

14.9.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

Figure 91. DMEM Control Register 0 (DMEMCTL0)

DMEM BASE ADDR																0	DBYPCB	DBYPAT	DBYPDEC	DECUE	0	DBAPD	DPEIE	DICWE	DSWCE	DDAUEC	DCPECE	DSECE
----------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	---	--------	--------	---------	-------	---	-------	-------	-------	-------	--------	--------	-------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

DCR 496; Read/Write; Reset⁽¹⁾: 19'b0 || p_dmem_rstcfg[8,6:7,0] || 4'b0 || p_dmem_rstcfg[1:5]; Privileged

1. Reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 177. DMEMCTL0 field descriptions

Bits	Name	Description
0:15	DMEM BASE ADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit. Note: Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that no cached entries are present in the D-Cache for the memory space to be occupied by the DMEM.
16:18	—	Reserved
19	DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (lbcbx, lhcbx, lwc bx, and stwtwx, sthwtx, stbwtx) accesses that target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port. 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[8] input.</i>
20	DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems that support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Atomic CPU accesses do not bypass the DMEM CPU port. 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[6] input.</i>

Table 177. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
21	DBYPDEC	DMEM Bypass Decorated CPU accesses DBYPDEC can be used to force decorated accesses (lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb) to the DMEM address space to be presented to external decoration hardware for systems that support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port. 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[7] input.</i>
22	DECUE	DMEM Error Correction Update Enable DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM. 0 Error Correction Update is disabled. 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be rewritten with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.) <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[0] input.</i>
23	—	Reserved
24	DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port (p_dmem_baseaddr[0:n]) to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port (p_dmem_baseaddr[0:n]) is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port (p_dmem_baseaddr[0:n]) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25	DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled. 1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits (p_dchk[0:1]).
26	DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set. 0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).

Table 177. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
27:28	DSWCE	DMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte writes) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors. <i>Note: This field is updated on m_por and slv_reset_b based on the values of the p_dmem_rstcfg[1:2] inputs.</i>
29	DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[3] input.</i>
30	DCPECE	DMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[4] input.</i>
31	DSECE	DMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[5] input.</i>

14.9.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses that target DMEM address range, regardless of whether access may bypass the DMEM based on settings in DMEMCTL0_{DBYPCB}, DBYPATF, DBYPDEC.

Figure 92. DMEM Control Register 1 (DMEMCTL1)

DSWAC3	DSWAC2	DSWAC1	DSWAC0	DSRAC3	DSRAC2	DSRAC1	DSRAC0	DUWAC3	DUWAC2	DUWAC1	DUWAC0	DURAC3	DURAC2	DURAC1	DURAC0																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 498; Read/Write; Reset⁽¹⁾: 32'b0; Privileged

1. Reset by **m_por** and **p_reset_b**.

Table 178. DMEMCTL1 field descriptions

Bits	Name	Description
0:1	DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11</i> <i>Note:</i>
2:3	DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10.</i>
4:5	DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01.</i>

Table 178. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
6:7	DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2'b00.</i>
8:9	DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2'b11</i> <i>Note:</i>
10:11	DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2'b10.</i>
12:13	DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2'b01.</i>

Table 178. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
14:15	DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2'b00.</i>
16:17	DUWAC3	DMEM User Write Access Control for Quadrant 3 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2'b11.</i> <i>Note:</i>
18:19	DUWAC2	DMEM User Write Access Control for Quadrant 2 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2'b10.</i>
20:21	DUWAC1	DMEM User Write Access Control for Quadrant 1 00 User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2'b01.</i>

Table 178. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
22:23	DUWAC0	DMEM User Write Access Control for Quadrant 0 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00.</i>
24:25	DURAC3	DMEM User Read Access Control for Quadrant 3 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved <i>Note: Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11.</i> <i>Note:</i>
26:27	DURAC2	DMEM User Read Access Control for Quadrant 2 <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10</i> 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved

Table 178. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
28:29	DURAC1	DMEM User Read Access Control for Quadrant 1 <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01</i> 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved
30:31	DURAC0	DMEM User Read Access Control for Quadrant 0 <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00</i> 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved

14.9.2.4 IMEM Configuration Register 0 (IMEMCFG0)

The IMEM Configuration Register 0 (IMEMCFG0) is a 32-bit read-only register. IMEMCFG0 provides information about the configuration of the e200z420n3 local instruction memory design. The contents of the IMEMCFG0 register can be read using a **mf spr** instruction. The SPR number for IMEMCFG0 is 695 in decimal. The IMEMCFG0 register is shown in [Figure 93](#).

Figure 93. IMEM Configuration Register 0 (IMEMCFG0)

IMEM BASE ADDR																				0	IECUA	IECA	0	IMSIZE				0			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CURRENT IMEM BASE ADDRESS VALUE																				0	1	1	0	000100 (16 KB)				00			
SPR 695; Read-only; Privileged																															

The IMEMCFG0 fields are described in [Table 179](#).

Table 179. IMEMCFG0 field descriptions

Bits	Name	Description
0:19	IMEM BASE ADDR	IMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the IMEM. This field reflects the value of the external IMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of IMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. <i>Note: Low order bits of this field are driven to 0's when the IMEM size is > 4 KB</i>
20	—	Reserved
21	IECUA	IMEM Error Correction Update Available IECUA indicates an error scrubbing function for the IMEM is available. 0 Error Correction Update is not available 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access
22	IECA	IMEM Error Correction Available IECA indicates an error correction function for the IMEM is available. 0 Error Correction is not available 1 Error Correction is available
23	—	Reserved
24:29	IMSIZE	IMEM Size 000100 The size of the IMEM is 16 KB
30:31	—	Reserved

14.9.2.5 IMEM Control Register 0 (IMEMCTL0)

The IMEM Control Register 0 (IMEMCTL0) controls operation of certain functions of the IMEM logic.

Figure 94. IMEM Control Register 0 (IMEMCTL0)

IMEM BASE ADDR																		0	0	IECUE	0	IBAPD	0	ISWCE	IDAUEC	ICPECE	ISECF				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 497; Read/Write; Reset⁽¹⁾: 22'b0 ||| p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5]; Privileged

1. Reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 180. IMEMCTL0 field descriptions

Bits	Name	Description
0:17	IMEM BASE ADDR	IMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the IMEM when the external base address port inputs are disabled via the BAPD control bit. <i>Note: Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that the BTB is invalidated, and that no cached entries are present in the I-Cache for the memory space to be occupied by the IMEM.</i>
18:19	—	Reserved for Base Address extension
20:21	—	Reserved
22	IECUE	IMEM Error Correction Update Enable IECUE can be used to provide an error scrubbing function. This field is updated on m_por and slv_reset_b based on the value of the p_imem_rstcfg[0] input. 0 Error Correction Update is disabled 1 Error Correction Update is enabled. A correctable single-bit error detected on CPU instruction fetches or slave port read cycles will cause the IMEM to be re-written with the corrected data if the corresponding error checking enable bit is set (ICPECE for CPU accesses, ISECE for slave port accesses). IECUE can be used in conjunction with ICPECE and ISECE to provide an error scrubbing function.
23	—	Reserved
24	IBAPD	IMEM Base Address Port Disable This bit controls usage of the external Base Address port (p_imem_baseaddr[0:n]) to define the base address of the IMEM for CPU port accesses. It has no effect on IMEM slave port accesses. 0 The external IMEM base address port (p_imem_baseaddr[0:n]) is used to define the base address of the IMEM for CPU accesses. 1 The external IMEM base address port (p_imem_baseaddr[0:n]) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the IMEM for CPU accesses.
25:26	—	Reserved
27:28	ISWCE	IMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the IMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 4, 5, 6, or 7 byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. <i>Note: This field is updated on m_por and slv_reset_b based on the values of the p_imem_rstcfg[1:2] inputs.</i>

Table 180. IMEMCTL0 field descriptions (continued)

Bits	Name	Description
29	IDAUEC	IMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used. for checkbit/syndrome generation for IMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_imem_rstcfg[3] input.</i>
30	ICPECE	IMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_imem_rstcfg[4] input.</i>
31	ISECE	IMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed during read-modify-write operations for partial-width writes with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a Slave port access causes a bus error ERROR response to be generated. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_imem_rstcfg[5] input.</i>

14.9.2.6 IMEM Control Register 1 (IMEMCTL1)

The IMEM Control Register 1 (IMEMCTL1) provides protection functions for the IMEM. Separate Supervisor-mode and User-mode instruction fetch access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of IMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings.

Figure 95. IMEM Control Register 1 (IMEMCTL1)

0								ISXAC3		ISXAC2		ISXAC1		ISXAC0		0								IUXAC3		IUXAC2		IUXAC1		IUXAC0	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 499; Read/Write; Reset⁽¹⁾: 32'b0; Privileged

1. Reset by **m_por** and **p_reset_b**.

Table 181. IMEMCTL1 field descriptions

Bits	Name	Description
0:7	—	Reserved
8:9	ISXAC3	IMEM Supervisor Instruction Fetch Access Control for Quadrant 3 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC3 is used when these two bits = 2'b11.</i>
10:11	ISXAC2	IMEM Supervisor Instruction Fetch Access Control for Quadrant 2 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC2 is used when these two bits = 2'b10.</i>
12:13	ISXAC1	IMEM Supervisor Instruction Fetch Access Control for Quadrant 1 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC1 is used when these two bits = 2'b01.</i>
14:15	ISXAC0	IMEM Supervisor Instruction Fetch Access Control for Quadrant 0 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC0 is used when these two bits = 2'b00.</i>
16:23	—	Reserved
24:25	IUXAC3	IMEM User Instruction Fetch Access Control for Quadrant 3 00 User-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC3 is used when these two bits = 2'b11.</i>

Table 181. IMEMCTL1 field descriptions (continued)

Bits	Name	Description
26:27	IUXAC2	IMEM User Instruction Fetch Access Control for Quadrant 2 00 User-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC2 is used when these two bits = 2'b10.</i>
28:29	IUXAC1	MEM User Instruction Fetch Access Control for Quadrant 1 00 User-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC1 is used when these two bits = 2'b01.</i>
30:31	IUXAC0	IMEM User Instruction Fetch Access Control for Quadrant 0 00 User-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC0 is used when these two bits = 2'b00.</i>

14.10 End-to-End ECC support

This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

14.10.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

14.10.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

Figure 96. e2eECC Control Register 0 (E2ECTL0)

0																												IDAC	IECCEN	DDAC	DECCEN
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 510; Read/Write; Reset: 0x0 || p_e2e_rstcfg[0:3]; Privileged

Table 182. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. <i>Note: This field is updated on p_reset_b based on the value of the p_e2e_rstcfg[0] input.</i>
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. <i>Note: This field is updated on p_reset_b based on the value of the p_e2e_rstcfg[1] input.</i>
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. <i>Note: This field is updated on p_reset_b based on the value of the p_e2e_rstcfg[2] input.</i>
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. <i>Note: This field is updated on p_reset_b based on the value of the p_e2e_rstcfg[3] input.</i>

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the

initialization, software may enable e2eECC operation by writing the appropriate values to the E2ECTLO_{[I,D]ECCEN} control bits.

The E2ECTLO_{[I,D]DAC} control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

14.10.1.2 End-to-End ECC error generation by software

Error generation provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus (but not on cache hits).

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

Figure 97 shows the layout of E2EECSR0.

Figure 97. e2eECC Error Control/Status Register 0 (E2EECSR0)

0		RCHKBIT										0		WRC	INVC	0		WCHKBIT/CHKINVT													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 511; Read/Write; Reset: 0x0; Privileged

Table 183. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to p_hrchkbit[7:0] for external reads and to p_dchk[0:7] for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	—	Reserved

Table 183. E2EECSR0 field descriptions (continued)

Bits	Name	Description
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving p_hwchkbit[7:0] with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving p_hwchkbit[7:0] with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved <i>Note: Checkbit substitution has priority over Error injection.</i>
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying p_hwchkbit[7:0] based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying p_hwchkbit[7:0] based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved <i>Note: Checkbit substitution has priority over Error injection.</i>
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to p_hwchkbit[7:0]. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on p_hwchkbit[7:0] when error injection occurs.

15 Core e200z425n3 description

15.1 Overview of the e200z425n3 core

The e200z425n3 is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z425n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

A Lightweight Signal Processing Extension (LSP) APU is provided to support real-time SIMD fixed-point embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). The GPRs support register pairing in order to support vector instructions defined by the LSP APU. These instructions operate on a vector pair of 16-bit or 32-bit data types, and deliver vector and scalar results. A wide variety of data manipulation, multiply, multiply/accumulate, and dot product instructions along with specialized memory access instructions allow a sustained throughput of up to two 16×16 -bit multiply/accumulate operations per cycle.

The e200z425n3 processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

e200z425n3 contains an 8 KB Instruction Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit that protects various instruction and data memory areas is also included.

15.2 Register model

This section describes the registers implemented in the e200z425n3 core. The *PowerISA 2.06* architecture defines register-to-register operations for all computational instructions.

e200z425n3 extends usage of the general purpose registers to support EFPU/LSP APU operations on the 32-bit GPR registers.

[Figure 98](#) and [Figure 99](#) show the complete e200z425n3 register set, all of which are accessible in supervisor mode. [Figure 100](#) and [Figure 101](#) show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

Figure 98. e200z425n3 Supervisor Mode Programmer's Model SPRs

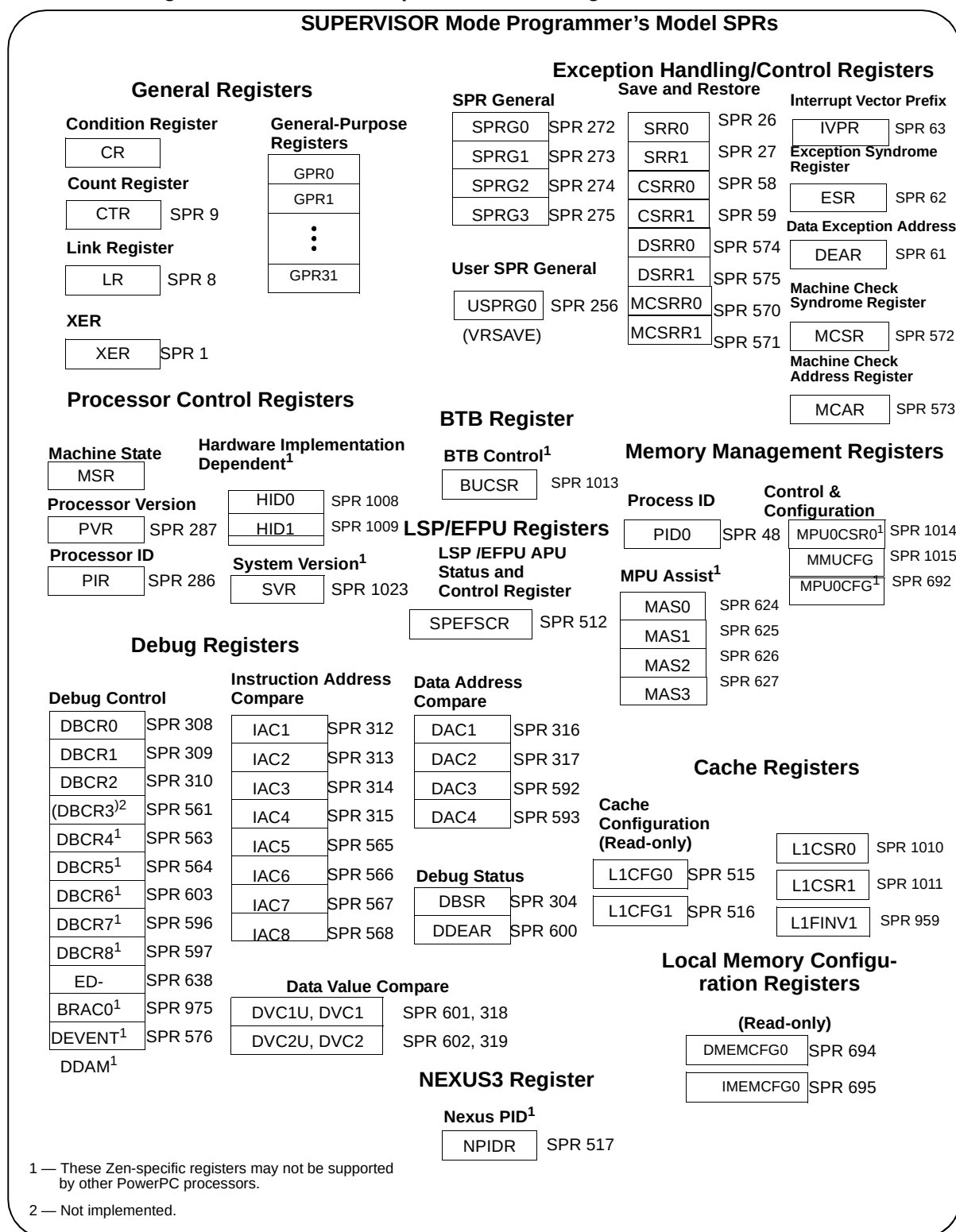


Figure 99. e200z425n3 Supervisor Mode Programmer's Model DCRs and PMRs

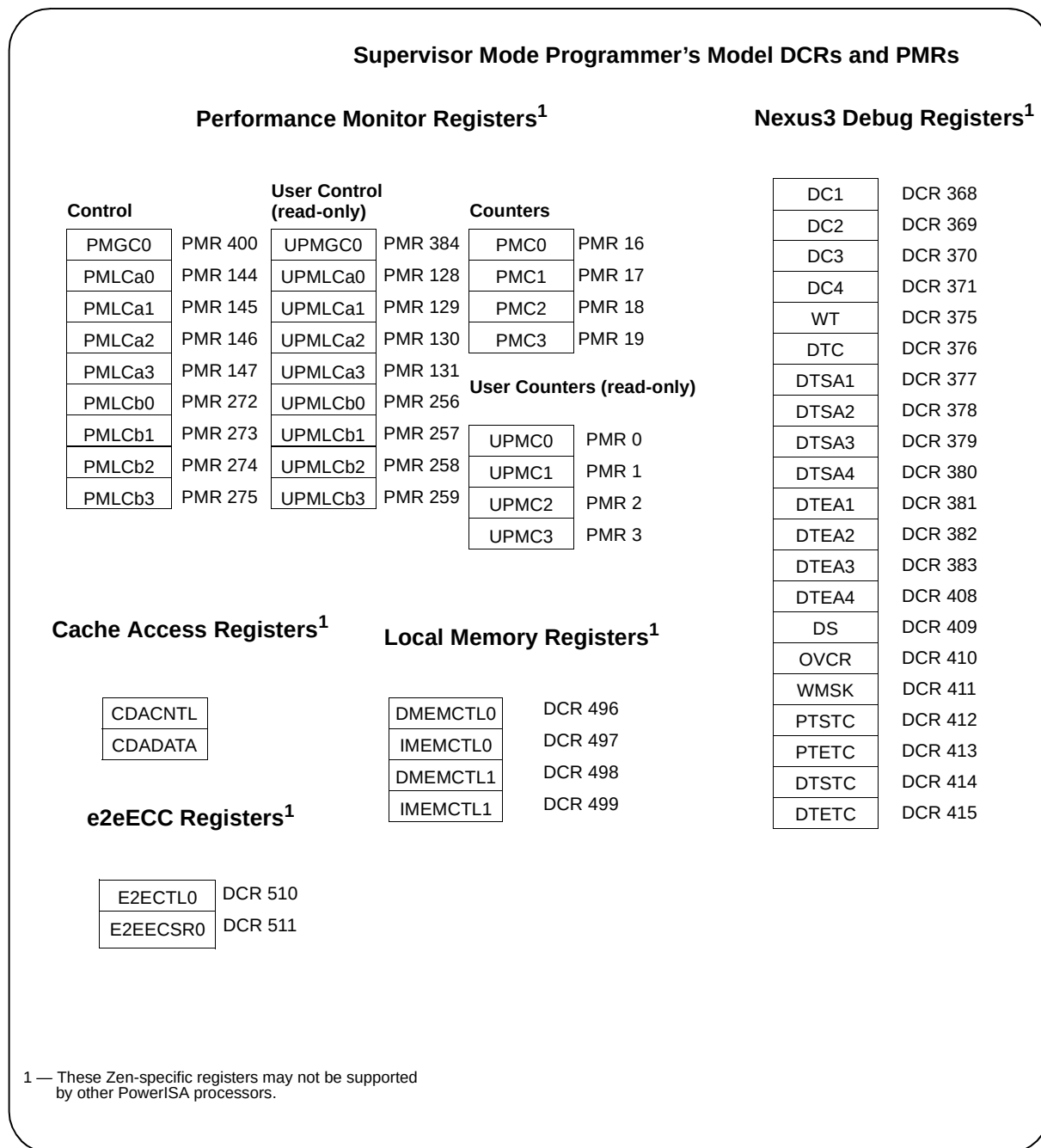
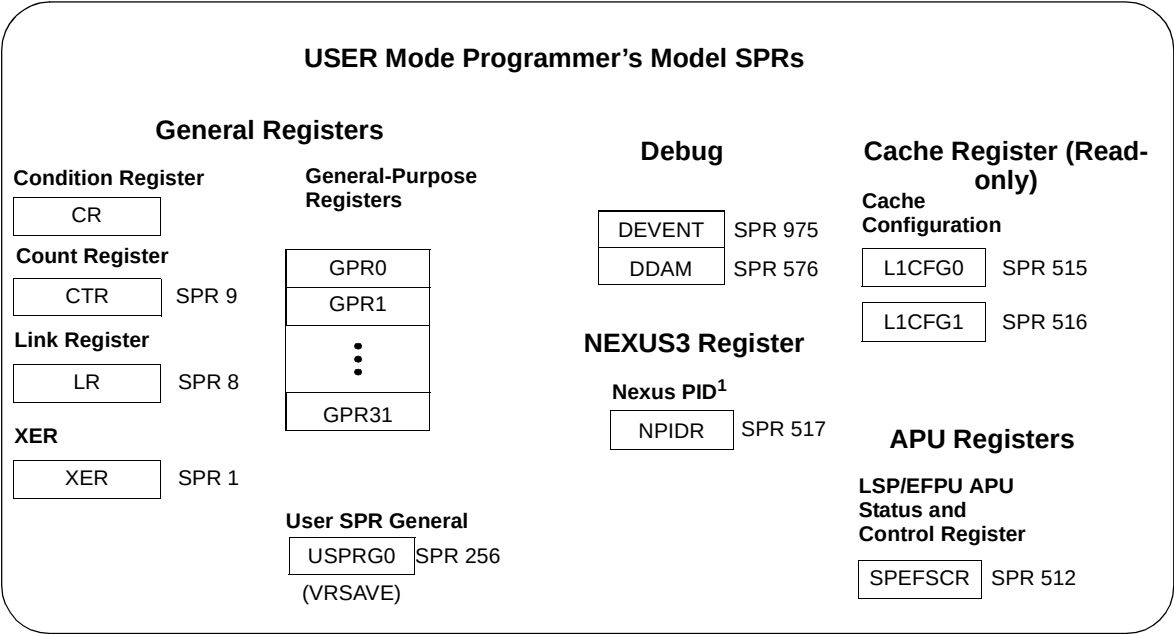


Figure 100. e200z425n3 User Mode Programmer's Model SPRs



Cache Register (Read-only)

Cache Configuration

L1CFG0

 SPR 515

L1CFG1

 SPR 516

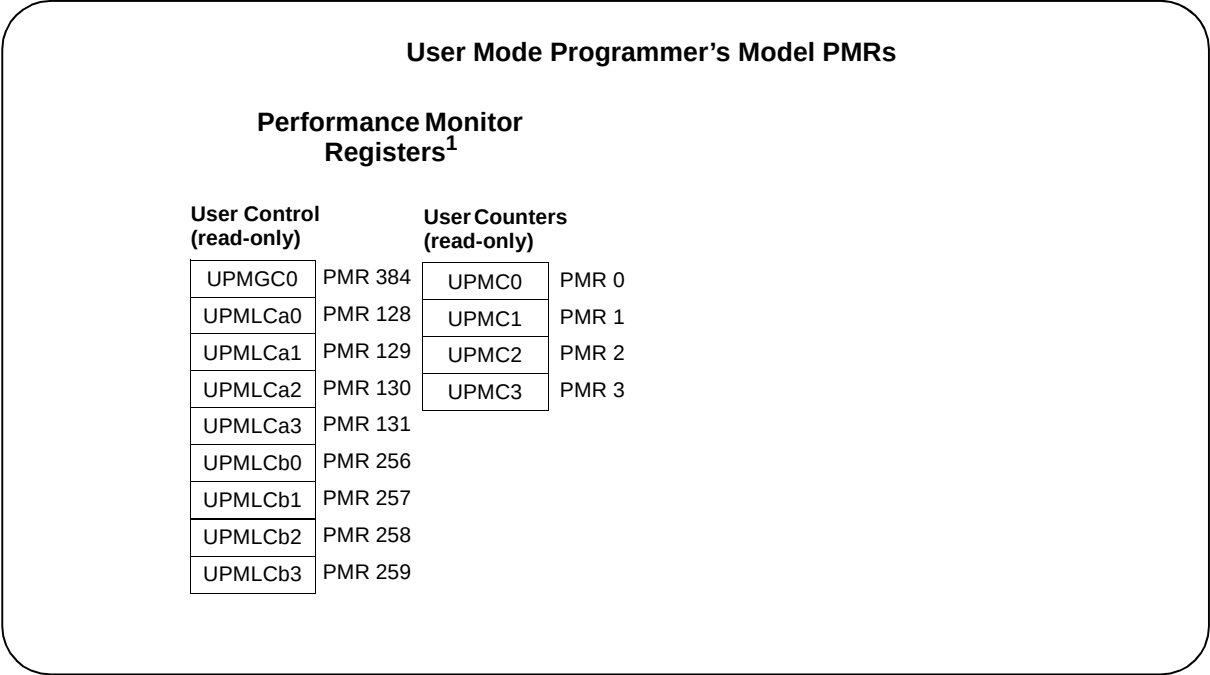
SPEFSCR

User SPR General

USPRG0

 SPR 256 (VRSAVE)

Figure 101. e200z425n3 User Mode Programmer's Model PMRs



User Counters (read-only)

UPMC0

 PMR 0

UPMC1

 PMR 1

UPMC2

 PMR 2

UPMC3

 PMR 3

15.3 Dual-issue operation

The instruction issue unit attempts to issue a pair of instructions to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the

operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions.

Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

[Table 184](#) shows the e200z425n3 concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue of those instructions.

Table 184. Concurrent instruction issue capabilities for e200z425n3

Class of instruction	Branch	Load/Store	Scalar integer	Scalar float	Vector integer non-MAC	Vector integer MAC	Special
Branch	—	yes	yes	yes	yes	yes	—
Load/Store	yes	—	yes	yes	yes	yes	—
Scalar integer	yes	yes	yes ⁽¹⁾	yes	yes	yes ⁽²⁾	—
Scalar float	yes	yes	yes	—	yes	yes	—
Vector integer non-MAC	yes	yes	yes	yes	yes	yes	—
Vector integer MAC	yes	yes	yes ⁽²⁾	yes	yes	—	—
Special	—	—	—	—	—	—	—

1. Excludes multiply or divide class instructions occurring in both issue slots.

2. Excludes vector MAC/multiply class instructions occurring with scalar multiply, or divide class instructions.

15.4 Reservation instructions and cache interactions

The CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure.

15.5 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. This register is also used by the LSP APU. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the **mf spr** and **mt spr** instructions. The SPEFSCR is shown in Figure 102.

Figure 102. LSP/EFPU Status and Control Register (SPEFSCR)

0										FINXS	FINVS	FDBZS	FUNFS	FOVFS	MODE	SOV	OV	FG	FX	FINV	FDBZ	FUNF	FOVF	0	FINXE	FINVE	FDBZE	FUNFE	FOVFE	FRMC	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 512: Read/Write; Reset: 0x0

The SPEFSCR bits are defined in [Table 185](#).

Table 185. LSP/EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a mtspr instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z425n3 supports only mode 0. Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by Zen)
16 (48)	SOV	Summary integer overflow Defined by LSP APU.
17 (49)	OV	Integer overflow Defined by LSP APU.

Table 185. LSP/EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler. FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the I dividend is a finite nonzero number.
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact ($FG \vee FX = 1$), or if the result of a Floating-point instruction does result in overflow ($FOVF = 1$) but Floating-point Overflow exceptions are disabled ($FOVFE = 0$), or if the result of a Floating-point instruction results in underflow ($FUNF = 1$ or $FUNFH = 1$) but Floating-point Underflow exceptions are disabled ($FUNFE = 0$), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled

Table 185. LSP/EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 (62:63)	FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

15.6 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

15.6.1 Cache overview

The e200z425n3 processor supports an 8 KB 2-way set-associative instruction cache with a 32-byte line size.

The cache improves system performance by providing low-latency data to the e200z425n3 instruction pipeline, which decouples processor performance from system memory performance.

The e200z425n3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

15.6.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of a data cache and for disabling ways in both caches. L1CSR0 implements the WID control field for instruction cache way disabling. The other fields are reserved. The L1CSR0 register is accessed using a **mfsp** or **mtsp** instruction. The SPR number for L1CSR0 is 1010 in decimal. The L1CSR0 register is shown in [Figure 103](#).

Figure 103. L1 Cache Control and Status Register 0 (L1CSR0)

WID	0																														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 1010; Read/Write; Reset: 0x0

The L1CSR0 fields are described in [Table 186](#).

Table 186. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:31	—	Reserved ⁽¹⁾

1. These bits are not implemented and should be written with zero for future compatibility.

15.6.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mf spr** or **mt spr** instruction. The SPR number for L1CSR1 is 1011 in decimal. The L1CSR1 register is shown in [Figure 104](#).

Figure 104. L1 Cache Control and Status Register 1 (L1CSR1)

0															ICECE	ICEI	ICLOC	0							ICEA	ICLOINV	0	ICABT	ICINV	ICE	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 1011: Read/Write: Reset: 0x0

The L1CSR1 fields are described in [Table 187](#).

Table 187. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved ⁽¹⁾
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled

Table 187. L1CSR1 field descriptions (continued)

Bits	Name	Description
16	ICEI	Instruction Cache Error Injection Enable ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits (p_chk[0:1]).
17:18	ICLOC	Instruction Cache Lockout Control <i>Note: For the icbi instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled.</i> <i>Note: When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored.</i> 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved ⁽¹⁾
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved ⁽¹⁾

Table 187. L1CSR1 field descriptions (continued)

Bits	Name	Description
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

15.6.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z425n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfsprr** instruction. The SPR number for L1CFG0 is 515 in decimal. The L1CFG0 register is shown in [Figure 105](#).

Figure 105. L1 Cache Configuration Register 0 (L1CFG0)

CARCH	CWPA	CFAHA	DCFISWA	0	DCBSIZE	DCREPL	DCLA	DCECA	DCNWAY	DCSIZE
-------	------	-------	---------	---	---------	--------	------	-------	--------	--------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1 0

SPR 515; Read-only

The L1CFG0 register indicates a cache architecture value (CARCH) of 2'b10, indicating that an I-Cache but no D-Cache is present.

15.6.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z425n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfsprr** instruction. The SPR number for L1CFG1 is 516 in decimal. The L1CFG1 register is shown in [Figure 106](#).

Figure 106. L1 Cache Configuration Register 1 (L1CFG1)

0	ICFISWA	0	ICBSIZE	ICREPL	ICLA	ICECA	ICNWAY	ICSIZE
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31								
0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0								

SPR 516; Read-only

The L1CFG1 fields are described in [Table 188](#).

Table 188. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved — read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved — read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 The size of the instruction cache is 8 KB

15.6.6 Cache Invalidate by Set and Way

e200z425n3 supports cache set/way invalidation under software control. The instruction cache may be invalidated by index and way through a **mtspr l1finv1** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV1) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when the instruction cache is disabled.

15.6.6.1 L1FINV1

The SPR number for L1FINV1 is 959 in decimal. The L1FINV1 register is shown in [Figure 107](#).

Figure 107. L1 Flush/Invalidate Register 1 (L1FINV1)

0	CWAY	0	CSET	0	CCMD
---	------	---	------	---	------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 959; Read/Write; Reset: 0x0

The L1FINV1 bits are described in [Table 189](#).

Table 189. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved ⁽¹⁾ for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8:19	—	Reserved ⁽¹⁾ for set extension
20:26	CSET	Cache Set Specifies the instruction cache set to be selected
27:28	—	Reserved ⁽¹⁾ for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

15.6.7 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of the instruction cache. Seven parity check bits are provided for each tag entry for the tag array to support error detection and correction (ECC — SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, which are used for single- and double-bit error detection (EDC — DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the L1CSR1_{ICECE} control field. When error checking is enabled, checking is performed on each cache access. Errors are not

signaled by the instruction cache when cache error checking is disabled ($L1CSR1_{ICECE} = 0$).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

15.6.7.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the cache, each cache line in the I-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by $L1CSR1_{ICLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on instruction fetch accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via $L1CSR1_{ICLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, cache contents are ignored for lines with those errors. Cache invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable access will perform the lockout function if required.

15.6.8 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction cache.

Error injection into the instruction cache operates as follows:

- If L1CSR1_{ICEI} is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi** instruction, or an L1FINV1 invalidation operation.

15.7 Exceptions

Interrupts implemented in e200z425n3 and the exception conditions that cause them are listed in [Table 190](#).

Table 190. Exceptions and conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] 2'b00	Reset
Critical Input	0x00 ⁽¹⁾	p_critint_b is asserted and MSR _{CE} = 1.
Machine check	0x10	1. p_mcp_b transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. Parity Error signaled on Local Memory access 5. External bus error 6. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40 ⁽²⁾	Interrupt Controller interrupt and MSR _{EE} = 1
Alignment	0x50	1. lmw, stmw not word aligned 2. lwarx or stwcx. not word aligned, lharx or sthcx. not halfword aligned 3. LSP ld and st instructions not properly aligned 4. dcbz
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/PMGC0 _{UDI} = 0
System call	0x80	Execution of the System Call (se_sc) instruction

Interrupt type	Interrupt vector offset value	Causing conditions
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/PMGC0 _{UDI} = 1, MPU
EFPU Data Exception	0xA0	See section “Embedded Floating-point Data Exception” in the Power Architecture core reference manual relative to this core.
EFPU Round Exception	0xB0	See section “Embedded Floating-point Round Exception” in the Power Architecture core reference manual relative to this core.
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectorized Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectorized External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.

0				PIL	PpR	Ptr	FP	ST	0		0	0	BO	0	0								SPV	0	VLEMI	0		0	0		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 62: Read/Write: Reset: 0x0

The ESR bits are defined in [Table 191](#).

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For core e200z425n3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—

Table 191. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	LSP/EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

15.7.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z425n3 MSR is shown in [Figure 109](#).

Figure 109. Machine State Register (MSR)

0														CE	0	EE	PR	0	ME	0	0	DE	0										PMM	RI	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				

Read/ Write; Reset: 0x0

The MSR bits are defined in [Table 192](#).

Table 192. MSR field descriptions

Bit(s)	Name	Description
0:13	—	Reserved
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved

Table 192. MSR field descriptions (continued)

Bit(s)	Name	Description
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	—	Reserved
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	—	Reserved
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23:28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR _{PR} and MSR _{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLC _{an} Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

15.7.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in [Figure 110](#).

Figure 110. Machine Check Syndrome Register (MCSR)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																										
MCP		IC_DPERR		0		DC_DPERR		EXCP_ERR		IC_TPERR		DC_TPERR		IC_LKERR		DC_LKERR		0		NMI		MAV		MEA		U		IF		LD		ST		G		0		STACK_ERR		IMEM_PERR		DMEM_RDPERR		DMEM_WRPERR		0		BUS_IRERR		BUS_DRERR		BUS_WRERR		BUS_WRDSI		0	

SPR 572; Read/Clear; Reset: 0x0

[Table 193](#) describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as “write 1 to clear.” Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once MSR_{ME} = 1.

Table 193. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type ⁽¹⁾	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR ⁽²⁾	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR ⁽²⁾	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR ⁽²⁾	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—

Table 193. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type ⁽¹⁾	Recoverable
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1, MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address ⁽³⁾	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise

Table 193. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type ⁽¹⁾	Recoverable
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.

— “Error Report” indicates that this bit is only set for error report exceptions that cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

— “Status” indicates that this bit provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

— “NMI” indicates that this bit is only set for the non-maskable interrupt type exception, which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a ‘1’ to the bit position.

— “Async Mchk” indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any “Async Mchk” bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME} = 1. If MSR_{ME} = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

2. Will not be set by e200z425n3 since no data cache is present.

3. Note that since no address translation mechanism is present on e200z425n3, the MEA value is always set to ‘0’.

15.7.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in [Figure 111](#).

Figure 111. Zen Interrupt Vector Prefix Register (IVPR)

Vector Base	0
-------------	---

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 63; Read/Write

The IVPR fields are defined in [Table 194](#).

Table 194. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

15.7.5 Interrupt Definitions

15.7.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE} , the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

[Table 195](#) lists register settings when a Critical Input interrupt is taken.

Table 195. Critical Input Interrupt—register settings

Register	Setting description		
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
CSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME — FE0 0 DE —/0 ⁽¹⁾	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	IVPR _{0:23} 0x00 (autovectored) IVPR _{0:15} (IVPR _{16:29} p_voffset[0:13]) 2'b00 (non-autovectored)		

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register (HID0_{CICLRDE}).

15.7.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register (HID0_{MCCLRDE}).

When a Machine Check interrupt is taken, registers are updated as shown in [Table 196](#).

Table 196. Machine Check Interrupt—register settings

Register	Setting description			
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.			
MCSRR1	Set to the contents of the MSR at the time of the interrupt			
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0/— ⁽¹⁾	FE1 0 IS 0 DS 0 PMM 0 RI 0	
ESR	Unchanged			
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.			
Vector	IVPR _{0:23} 0x10			

1. Clearing of DE is optionally supported by control in HID0.

15.7.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

[Table 197](#) lists register settings when a DSI is taken.

Table 197. Data Storage Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting load/store instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Access:	[ST], [SPV], VLEMI. All other bits cleared.	
MCSR	Unchanged		

Table 197. Data Storage Interrupt—register settings (continued)

Register	Setting description
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.
Vector	IVPR _{0:23} 0x20

15.7.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

[Table 198](#) lists register settings when an ISI is taken.

Table 198. Instruction Storage Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	IVPR _{0:23} 0x30		

15.7.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

[Table 199](#) lists register settings when an External Input interrupt is taken.

Table 199. External Input Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table 199. External Input Interrupt—register settings (continued)

Register	Setting description			
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —	
ESR	Unchanged			
MCSR	Unchanged			
DEAR	Unchanged			
Vector	IVPR _{0:23} 0x40 (autovectored) IVPR _{0:15} (IVPR _{16:29} p_voffset[0:13]) 2'b00 (non-autovectored)			

15.7.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.
- Execution of an LSP APU load or store instruction that is not properly aligned.

[Table 200](#) lists register settings when an alignment interrupt is taken.

Table 200. Alignment Interrupt—register settings

Register	Setting description			
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.			
SRR1	Set to the contents of the MSR at the time of the interrupt			
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —	
ESR	[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged			
DEAR	Set to the effective address of a byte of the load or store access causing the violation.			
Vector	IVPR _{0:23} 0x50			

15.7.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z425n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspir** instructions with an undefined SPR specified
- **mtdcr** and **mfddcr** instructions with an undefined DCR specified

The e200z425n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when $MSR_{PR} = 1$ (user mode):

- A privileged instruction
- **mtspr** and **mfspir** instructions that specify a SPRN value with $SPRN_5 = 1$ (even if the SPR is undefined)

The e200z425n3 will invoke a Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfddcrx**, **mtddcrx**, or on any *PowerISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

[Table 201](#) lists register settings when a Program interrupt is taken.

Table 201. Program Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Illegal, Unimplemented: Privileged: Trap:	PIL, VLEMI. All other bits cleared. PPR, VLEMI. All other bits cleared. PTR, VLEMI. All other bits cleared.	
MCSR	Unchanged		
DEAR	Unchanged		
Vector	IVPR _{0:23} 0x60		

15.7.5.8 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

[Table 202](#) lists register settings when a System Call interrupt is taken.

Table 202. System Call Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the instruction <i>following</i> the system call instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	IVPR _{0:23} 0x80		

15.7.5.9 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both DBCR0_{IDM} = 1 (internal debug mode) and MSR_{DE} = 1).

[Table 203](#) lists register settings when a Debug interrupt is taken.

Table 203. Debug Interrupt—register settings

Register	Setting description		
DSRR0 ⁽¹⁾	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP. Set to the effective address of the next instruction to be executed <i>following</i> the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
DSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE —/0 ⁽²⁾ EE —/0 ⁽²⁾ PR 0	FP 0 ME — FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI —

Table 203. Debug Interrupt—register settings (continued)

Register	Setting description	
DBSR ⁽³⁾	Unconditional Debug Event:	UDE
	Instr. Complete Debug Event:	ICMP
	Branch Taken Debug Event:	BRT
	Interrupt Taken Debug Event:	IRPT
	Critical Interrupt Taken Debug Event:	CIRPT
	Trap Instruction Debug Event:	TRAP
	Instruction Address Compare:	{IAC}
	Data Address Compare:	{DACR, DACW}
	Debug Notify Interrupt:	DNI
	Return Debug Event:	RET
	Critical Return Debug Event:	CRET
	External Debug Event:	{DEVT1, DEVT2}
	Performance Monitor Debug Event:	PMI
	MPU Debug Event:	MPU
	and optionally, an Imprecise Debug Event flag	{IDE}
ESR	Unchanged	
MCSR	Unchanged	
DEAR	Unchanged	
Vector	IVPR _{0:23} 0x90	

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

15.7.5.10 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

[Table 204](#) lists register settings when a System Reset interrupt is taken.

Table 204. System Reset Interrupt—register settings

Register	Setting description
CSRR0	Undefined
CSRR1	Undefined

Table 204. System Reset Interrupt—register settings (continued)

Register	Setting description			
MSR	SPV 0	FP 0	FE1 0	
	WE 0	ME 0	IS 0	
	CE 0	FE0 0	DS 0	
	EE 0	DE 0	PMM 0	
	PR 0		RI 0	
ESR	Cleared			
DEAR	Undefined			
Vector	[p_rstbase[0:29]] 2'b00			

15.7.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

[Table 205](#) lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 205. Embedded Floating-point Data Interrupt—register settings

Register	Setting description			
SRR0	Set to the effective address of the excepting EFPU instruction.			
SRR1	Set to the contents of the MSR at the time of the interrupt			
MSR	SPV 0	FP 0	FE1 0	
	WE 0	ME —	IS 0	
	CE —	FE0 0	DS 0	
	EE 0	DE —	PMM 0	
	PR 0		RI —	
ESR	SPV, VLEMI. All other bits cleared.			
MCSR	Unchanged			
DEAR	Unchanged			
Vector	IVPR _{0:23} 0xA0			

15.7.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

[Table 206](#) lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 206. Embedded Floating-point Round Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table 206. Embedded Floating-point Round Interrupt—register settings (continued)

Register	Setting description			
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —	
ESR	SPV, VLEMI. All other bits cleared.			
MCSR	Unchanged			
DEAR	Unchanged			
Vector	IVPR _{0:23} 0xB0			

15.7.5.13 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMCx register overflow condition occurs with the following settings:

- PMLCax_{CE} = 11; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, PMGC0_{PMIE} must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

[Table 207](#) lists register settings when an performance monitor interrupt is taken.

Table 207. Performance Monitor Interrupt—Register Settings

Register	Setting description			
SRR0/ DSRR0 ⁽¹⁾	Set to the effective address of the next instruction to be executed.			
SRR1/ DSRR1 ⁽¹⁾	Set to the contents of the MSR at the time of the interrupt.			
MSR	SPV 0 WE 0 CE —/0 ⁽²⁾ EE 0/— ⁽³⁾ PR 0	FP 0 ME — FE0 0 DE —/0 ⁽⁴⁾	FE1 0 IS 0 DS 0 PMM 0 RI —	
ESR	Unchanged			
MCSR	Unchanged			
DEAR	Unchanged			
Vector	IVPR _{0:23} 0x70			

1. DSRR0/1 are used if PMGC0_{UDI} = 1

2. CE is cleared if PMGC0_{UDI} = 1 and HID0_{DCLRC} = 1

3. EE is not cleared if $PMGC0_{UDI} = 1$ and $HID0_{DCLREE} = 0$
4. DE is cleared if $PMGC0_{UDI} = 1$

15.8 Memory Protection Unit (MPU)

15.8.1 MPU Overview

The e200z425n3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

15.8.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mf spr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

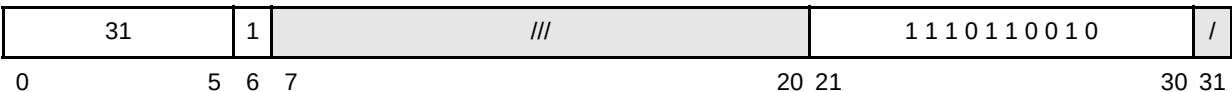
The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

15.8.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure
mpu read entry

mpure



```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

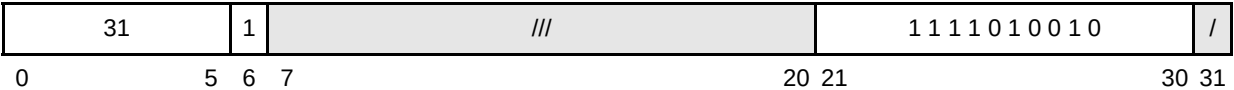
15.8.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe

mpu write entry

mpuwe



```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

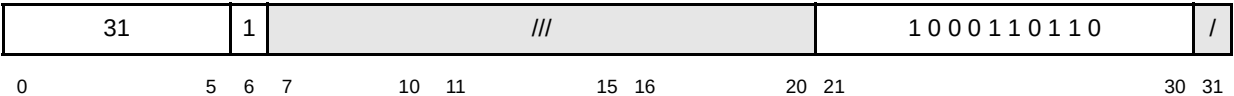
15.8.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the e200z425n3.

mpusync

MPU Synchronize

mpusync



15.8.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z425n3 MPU design. The MMUCFG register is shown in [Figure 112](#).

Figure 112. MMU/MPU Configuration Register (MMUCFG)

0	RASIZE	0	PIDSIZE	NMPUS	NTLBS	MAVN
---	--------	---	---------	-------	-------	------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 1015; Read-Only

The MMUCFG bits are described in [Table 208](#).

Table 208. MMUCFG field descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

15.8.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z425n3 MPU. The MPU0CFG register is shown in [Figure 113](#).

Figure 113. MPU0 Configuration Register (MPU0CFG)

FASSOC	0	MINSIZE	MAXSIZE	IPROT	0	UAMSKA	0	SHENTRY	DENTRY	IENTRY
--------	---	---------	---------	-------	---	--------	---	---------	--------	--------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 692; Read-Only

The MPU0CFG bits are described in [Table 209](#).

Table 209. MPU0CFG field descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15	MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16	IProt	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28	DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

15.8.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in [Figure 114](#).

Figure 114. MPU0 Control and Status Register 0 (MPU0CSR0)

0																BYPSR	BYPSW	BYPSX	BYPUR	BYPURW	BYPUX	0	DRDEN	DWDEN	IDEN	0	TIDCTL	0	MPUFI	MPUIEN	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 1014; Read/ Write; Reset: 0x0⁽¹⁾

- Reset by processor reset **p_reset_b** if EDBCR0_{EDM} = 0, as well as unconditionally by **m_por**. If EDBCR0_{EDM} = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Specifically, this applies to the DRDEN, DWDEN, and IDEN control bits.

The MPU0CSR0 bits are described in [Table 210](#).

Table 210. MPU0CSR0 field descriptions

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17	BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19	BYPUR	Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses
20	BYP UW	Bypass MPU protections for User Write accesses This bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21	BYP UX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
22:23	—	Reserved

Table 210. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
24	DRDEN	Data Read Debug Enable This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses
25	DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses 1 MPU debug events are enabled for data write accesses
26	IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses 1 MPU debug events are enabled for instruction accesses
27	—	Reserved
28	TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode
29	—	Reserved

Table 210. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
30	MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. 0 No flash invalidate 1 MPU1 invalidation operation <i>Note: Entries that have the IPROT bit set will not be invalidated.</i>
31	MPUEN	MPU Enable This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

15.8.9 MPU Assist Registers (MAS)

The e200z425n3 uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mf spr** and **mt spr** instructions.

The MAS0 register is shown in [Figure 115](#). Fields are defined in [Table 211](#).

Figure 115. MPU Assist Register 0 (MAS0)

VALID	IPROT	SEL		0	RO	DEBUG	INST	SHD	0		ESEL				0	UAMSK				UW	SW	UX/UR	SX/SR	IOVR	GOVR ⁽¹⁾	1	1	I	0	G ₁ ⁽	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 624; Read/ Write; Reset: Unaffected

1. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Table 211. MAS0—MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 This MPU entry is invalid 1 This MPU entry is valid
1	IPROT	Invalidation Protect 0 Entry is not protected from invalidation 1 Entry is protected from invalidation

Table 211. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
2:3	SEL	Selects MPU for access: 00 Reserved, no access 01 Reserved, no access 10 Select MPU 11 Reserved, no access
4	—	Reserved
5	RO	Read-Only When set to 1, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs. 0 This MPU entry is writable 1 This MPU entry is Read-only
6	DEBUG	Debug Control for Entry This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access. 0 This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access
7	INST	Instruction Entry When SHD = 0, this bit is used to select the INST entry portion of the region descriptor table for mpure and mpuwe instruction operations. When SHD = 1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. 0 This MPU entry is used for matching data accesses only 1 This MPU entry is used for matching instruction accesses only
8	SHD	Shared Entry Select This bit is used to control selection of the Shared portion of the region descriptor table. 0 The shared portion of the region descriptor table is not accessed on a mpure or mpuwe operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 The shared portion of the region descriptor table is accessed on a mpure or mpuwe operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected.
9:11	—	Reserved
12:15	ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved

Table 211. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
17:19	UAMSK	Upper Address Mask Control This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly. 000 No upper address bits are masked, address matching uses all 32 address bits for accesses 001 The most significant access address bit is forced to 0 before matching 010 The two most significant access address bits are forced to 0 before matching 011 The three most significant access address bits are forced to 0 before matching 100 The four most significant access address bits are forced to 0 before matching 101 The upper five access address bits are forced to 0 before matching 11x Reserved, do not use
20	UW	User Mode Write Permission Determines User mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 No User mode Write permission 1 User mode has Write permission
21	SW	Supervisor Mode Write / Read Permission Determines Supervisor mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 No Supervisor mode Write permission 1 Supervisor mode has Write permission
22	UX/UR	User Mode Execute / Read Permission Determines User mode Execute (X) permission when INST = 1, or User mode Read (R) permission when INST = 0. 0 No User mode Execute/ Read permission 1 User mode has Execute/ Read permission
23	SX/SR	Supervisor Mode Execute / Read Permission Determines Supervisor mode Execute (X) permission when INST = 1, or Supervisor mode Read (R) permission when INST = 0. 0 No Supervisor mode Execute/ Read permission 1 Supervisor mode has Execute/ Read permission
24	IOVR	Cache-Inhibit attribute Override Determines whether this matching entry overrides the I bit settings of other matching entries and the cache-inhibited region configuration signals. 0 No override of I attribute by entry 1 Entry I bit overrides I attribute
25	GOVR/—	G attribute Override Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 No override of G attribute by entry 1 Entry G bit overrides G attribute
26	—	Reserved
27	—	Reserved

Table 211. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
28	I	Cache Inhibited This attribute may be overridden by the cache-inhibited region configuration input settings. 0 This region is considered cacheable 1 This region is considered cache-inhibited
29	—	Reserved
30	G/—	Guarded This attribute may be overridden by the guarded region configuration input settings. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 All loads and stores to this region are performed without speculation (i.e. they are known to be required)
31	—	Reserved

The MAS1 register is shown in [Figure 116](#). Fields are defined in [Table 212](#).

Figure 116. MPU Assist Register 1 (MAS1)

0	TID	0	TIDMSK
---	-----	---	--------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 625; Read/Write; Reset: Unaffected

Table 212. MAS1—descriptor context and configuration control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.

Table 212. MAS1—descriptor context and configuration control

Bit	Name	Comments, or Function when Set
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx TID[0] comparison to PID0[0] not masked 1xxxxxxx TID[0] comparison to PID0[0] is masked x0xxxxxx TID[1] comparison to PID0[1] not masked x1xxxxxx TID[1] comparison to PID0[1] is masked xx0xxxxx TID[2] comparison to PID0[2] not masked xx1xxxxx TID[2] comparison to PID0[2] is masked xxx0xxxx TID[3] comparison to PID0[3] not masked xxx1xxxx TID[3] comparison to PID0[3] is masked xxxx0xxx TID[4] comparison to PID0[4] not masked xxxx1xxx TID[4] comparison to PID0[4] is masked xxxxx0xx TID[5] comparison to PID0[5] not masked xxxxx1xx TID[5] comparison to PID0[5] is masked xxxxxx0x TID[6] comparison to PID0[6] not masked xxxxxx1x TID[6] comparison to PID0[6] is masked xxxxxxx0 TID[7] comparison to PID0[7] not masked xxxxxxx1 TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in [Figure 117](#). Fields are defined in [Table 213](#).

Figure 117. MPU Assist Register 2 (MAS2)

UPPER BOUND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SPR 626; Read/ Write; Reset: Unaffected																															

Table 213. MAS2—Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND ≤ EA ≤ UPPER_BOUND

The MAS3 register is shown in [Figure 118](#). Fields are defined in [Table 214](#).

Figure 118. MPU Assist Register 3 (MAS3)

LOWER BOUND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SPR 627: Read/ Write: Reset: Unaffected																															

Table 214. MAS3—Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND ≤ UPPER_BOUND

15.8.10 MAS Registers Summary

The MAS registers are summarized in [Figure 119](#).

Figure 119. MPU Assist Registers Summary

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
MAS0	VALID	IPROT	SEL		0	RO	DEBUG	INST	SHD	0		ESEL			0	UAMSK			UW	SW	UX/UR	SX/SR	IOVR	GOVR ⁽¹⁾	1	1	I	0	G ⁽¹⁾	0		
MAS1	0								TID							0							TIDMSK									
MAS2	UPPER BOUND																															
MAS3	LOWER BOUND																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

1. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1.

15.9 Local memories

15.9.1 Local Instruction and Data memory overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added. This provides low latency access (zero wait-states) similar to an instruction or data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large capacity “tightly coupled” storage capability.

15.9.2 Local memory control and configuration

DMEM and IMEM local memory configuration information is provided by a set of privileged read-only SPRs that are accessed using **mf spr** instructions. DMEM and IMEM local memory operation is controlled by a set of privileged device control registers (DCRs) that are accessed using the **mf dcr** and **mt dcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

15.9.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z425n3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mf spr**

instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in [Figure 120](#).

Figure 120. DMEM Configuration Register 0 (DMEMCFG0)

DMEM BASE ADDR																				0	DECUA	DECA	0	DMSIZE					0		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CURRENT DMEM BASE ADDRESS VALUE																				0	1	1	0	001000 (32 KB)					00		
SPR 694; Read-only; Privileged																															

The DMEMCFG0 fields are described in [Table 215](#).

Table 215. DMEMCFG0 field descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. <i>Note: Low order bits of this field are driven to 0s when the DMEM size is > 4 KB</i>
20	—	Reserved
21	DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available. 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access.
22	DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available. 1 Error Correction is available.
23	—	Reserved
24:29	DMSIZE	DMEM Size 001000 The size of the DMEM is 32 KB.
30:31	—	Reserved

15.9.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

Figure 121. DMEM Control Register 0 (DMEMCTL0)

DMEM BASE ADDR	0	DBYPCB	DBYPAT	DBYPDEC	DECUE	0	DBAPD	DPEIE	DICWE	DSWCE	DDAUEC	DCPECE	DSECE
----------------	---	--------	--------	---------	-------	---	-------	-------	-------	-------	--------	--------	-------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

DCR 496; Read/Write; Reset⁽¹⁾: 19'b0 || p_dmem_rstcfg[8,6:7,0] || 4'b0 || p_dmem_rstcfg[1:5]; Privileged

1. Reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 216. DMEMCTL0 field descriptions

Bits	Name	Description
0:16	DMEM BASE ADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit.
17:18	—	Reserved
19	DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (lbcbx , lhcbx , lwc bx , and stwwtx , sthwt x , stbwt x) accesses that target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port. 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[8] input.</i>
20	DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems that support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Atomic CPU accesses do not bypass the DMEM CPU port. 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[6] input.</i>

Table 216. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
21	DBYPDEC	DMEM Bypass Decorated CPU accesses DBYPDEC can be used to force decorated accesses (lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb) to the DMEM address space to be presented to external decoration hardware for systems that support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port. 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[7] input.</i>
22	DECUE	DMEM Error Correction Update Enable DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM. 0 Error Correction Update is disabled. 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be rewritten with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.) <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[0] input.</i>
23	—	Reserved
24	DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port (p_dmem_baseaddr[0:n]) to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port (p_dmem_baseaddr[0:n]) is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port (p_dmem_baseaddr[0:n]) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25	DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled. 1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits (p_dchk[0:1]).
26	DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set. 0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).

Table 216. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
27:28	DSWCE	DMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte writes) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors. <i>Note: This field is updated on m_por and slv_reset_b based on the values of the p_dmem_rstcfg[1:2] inputs.</i>
29	DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[3] input.</i>
30	DCPECE	DMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[4] input.</i>
31	DSECE	DMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_dmem_rstcfg[5] input.</i>

15.9.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses that target DMEM address range, regardless of whether access may bypass the DMEM based on settings in DMEMCTL0_{DBYPCB}, DBYPATF, DBYPDEC.

Figure 122. DMEM Control Register 1 (DMEMCTL1)

DSWAC3		DSWAC2		DSWAC1		DSWAC0		DSRAC3		DSRAC2		DSRAC1		DSRAC0		DUWAC3		DUWAC2		DUWAC1		DUWAC0		DURAC3		DURAC2		DURAC1		DURAC0	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 498; Read/Write; Reset⁽¹⁾: 32'b0; Privileged

1. Reset by **m_por** and **p_reset_b**.

Table 217. DMEMCTL1 field descriptions

Bits	Name	Description
0:1	DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11</i> <i>Note:</i>
2:3	DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10.</i>
4:5	DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01.</i>

Table 217. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
6:7	DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2'b00.</i>
8:9	DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2'b11</i> <i>Note:</i>
10:11	DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2'b10.</i>
12:13	DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2'b01.</i>

Table 217. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
14:15	DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2'b00.</i>
16:17	DUWAC3	DMEM User Write Access Control for Quadrant 3 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2'b11.</i> <i>Note:</i>
18:19	DUWAC2	DMEM User Write Access Control for Quadrant 2 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2'b10.</i>
20:21	DUWAC1	DMEM User Write Access Control for Quadrant 1 00 User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2'b01.</i>

Table 217. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
22:23	DUWAC0	DMEM User Write Access Control for Quadrant 0 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00.</i>
24:25	DURAC3	DMEM User Read Access Control for Quadrant 3 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved <i>Note: Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11.</i> <i>Note:</i>
26:27	DURAC2	DMEM User Read Access Control for Quadrant 2 <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10</i> 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved

Table 217. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
28:29	DURAC1	DMEM User Read Access Control for Quadrant 1 <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01</i> 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved
30:31	DURAC0	DMEM User Read Access Control for Quadrant 0 <i>Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00</i> 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved

15.9.2.4 IMEM Configuration Register 0 (IMEMCFG0)

The IMEM Configuration Register 0 (IMEMCFG0) is a 32-bit read-only register. IMEMCFG0 provides information about the configuration of the e200z425n3 local instruction memory design. The contents of the IMEMCFG0 register can be read using a **mfsprr** instruction. The SPR number for IMEMCFG0 is 695 in decimal. The IMEMCFG0 register is shown in [Figure 123](#).

Figure 123. IMEM Configuration Register 0 (IMEMCFG0)

IMEM BASE ADDR																				0	IECUA	IECA	0	IMSIZE				0
----------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	---	-------	------	---	--------	--	--	--	---

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
 CURRENT IMEM BASE ADDRESS VALUE 0 1 1 0 000100 (16 KB) 00

SPR 695; Read-only; Privileged

The IMEMCFG0 fields are described in [Table 218](#).

Table 218. IMEMCFG0 field descriptions

Bits	Name	Description
0:19	IMEM BASE ADDR	IMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the IMEM. This field reflects the value of the external IMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of IMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. <i>Note: Low order bits of this field are driven to 0's when the IMEM size is > 4 KB</i>
20	—	Reserved
21	IECUA	IMEM Error Correction Update Available IECUA indicates an error scrubbing function for the IMEM is available. 0 Error Correction Update is not available 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access
22	IECA	IMEM Error Correction Available IECA indicates an error correction function for the IMEM is available. 0 Error Correction is not available 1 Error Correction is available
23	—	Reserved
24:29	IMSIZE	IMEM Size 000100 The size of the IMEM is 16 KB
30:31	—	Reserved

15.9.2.5 IMEM Control Register 0 (IMEMCTL0)

The IMEM Control Register 0 (IMEMCTL0) controls operation of certain functions of the IMEM logic.

Figure 124. IMEM Control Register 0 (IMEMCTL0)

IMEM BASE ADDR																		0	0	IECUE	0	IBAPD	0	ISWCE	IDAUEC	ICPECE	ISECF				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 497; Read/Write; Reset⁽¹⁾: 22'b0 ||| p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5]; Privileged

1. Reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 219. IMEMCTL0 field descriptions

Bits	Name	Description
0:17	IMEM BASE ADDR	IMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the IMEM when the external base address port inputs are disabled via the BAPD control bit. <i>Note: Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that the BTB is invalidated, and that no cached entries are present in the I-Cache for the memory space to be occupied by the IMEM.</i>
18:19	—	Reserved for Base Address extension
20:21	—	Reserved
22	IECUE	IMEM Error Correction Update Enable IECUE can be used to provide an error scrubbing function. This field is updated on m_por and slv_reset_b based on the value of the p_imem_rstcfg[0] input. 0 Error Correction Update is disabled 1 Error Correction Update is enabled. A correctable single-bit error detected on CPU instruction fetches or slave port read cycles will cause the IMEM to be re-written with the corrected data if the corresponding error checking enable bit is set (ICPECE for CPU accesses, ISECE for slave port accesses). IECUE can be used in conjunction with ICPECE and ISECE to provide an error scrubbing function.
23	—	Reserved
24	IBAPD	IMEM Base Address Port Disable This bit controls usage of the external Base Address port (p_imem_baseaddr[0:n]) to define the base address of the IMEM for CPU port accesses. It has no effect on IMEM slave port accesses. 0 The external IMEM base address port (p_imem_baseaddr[0:n]) is used to define the base address of the IMEM for CPU accesses. 1 The external IMEM base address port (p_imem_baseaddr[0:n]) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the IMEM for CPU accesses.
25:26	—	Reserved
27:28	ISWCE	IMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the IMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 4, 5, 6, or 7 byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. <i>Note: This field is updated on m_por and slv_reset_b based on the values of the p_imem_rstcfg[1:2] inputs.</i>

Table 219. IMEMCTL0 field descriptions (continued)

Bits	Name	Description
29	IDAUEC	IMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used. for checkbit/syndrome generation for IMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_imem_rstcfg[3] input.</i>
30	ICPECE	IMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_imem_rstcfg[4] input.</i>
31	ISECE	IMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed during read-modify-write operations for partial-width writes with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a Slave port access causes a bus error ERROR response to be generated. <i>Note: This field is updated on m_por and slv_reset_b based on the value of the p_imem_rstcfg[5] input.</i>

15.9.2.6 IMEM Control Register 1 (IMEMCTL1)

The IMEM Control Register 1 (IMEMCTL1) provides protection functions for the IMEM. Separate Supervisor-mode and User-mode instruction fetch access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of IMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings.

Figure 125. IMEM Control Register 1 (IMEMCTL1)

0								ISXAC3		ISXAC2		ISXAC1		ISXAC0		0								IUXAC3		IUXAC2		IUXAC1		IUXAC0	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 499; Read/Write; Reset⁽¹⁾: 32'b0; Privileged

1. Reset by **m_por** and **p_reset_b**.

Table 220. IMEMCTL1 field descriptions

Bits	Name	Description
0:7	—	Reserved
8:9	ISXAC3	IMEM Supervisor Instruction Fetch Access Control for Quadrant 3 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC3 is used when these two bits = 2'b11.</i>
10:11	ISXAC2	IMEM Supervisor Instruction Fetch Access Control for Quadrant 2 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC2 is used when these two bits = 2'b10.</i>
12:13	ISXAC1	IMEM Supervisor Instruction Fetch Access Control for Quadrant 1 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC1 is used when these two bits = 2'b01.</i>
14:15	ISXAC0	IMEM Supervisor Instruction Fetch Access Control for Quadrant 0 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC0 is used when these two bits = 2'b00.</i>
16:23	—	Reserved
24:25	IUXAC3	IMEM User Instruction Fetch Access Control for Quadrant 3 00 User-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC3 is used when these two bits = 2'b11.</i>

Table 220. IMEMCTL1 field descriptions (continued)

Bits	Name	Description
26:27	IUXAC2	IMEM User Instruction Fetch Access Control for Quadrant 2 00 User-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC2 is used when these two bits = 2'b10.</i>
28:29	IUXAC1	MEM User Instruction Fetch Access Control for Quadrant 1 00 User-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC1 is used when these two bits = 2'b01.</i>
30:31	IUXAC0	IMEM User Instruction Fetch Access Control for Quadrant 0 00 User-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved <i>Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC0 is used when these two bits = 2'b00.</i>

15.10 End-to-End ECC support

This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

15.10.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

15.10.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

Figure 126. e2eECC Control Register 0 (E2ECTL0)

0																												IDAC	IECCEN	DDAC	DECCEN
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 510; Read/Write; Reset: 0x0 || p_e2e_rstcfg[0:3]; Privileged

Table 221. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. <i>Note: This field is updated on p_reset_b based on the value of the p_e2e_rstcfg[0] input.</i>
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. <i>Note: This field is updated on p_reset_b based on the value of the p_e2e_rstcfg[1] input.</i>
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. <i>Note: This field is updated on p_reset_b based on the value of the p_e2e_rstcfg[2] input.</i>
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. <i>Note: This field is updated on p_reset_b based on the value of the p_e2e_rstcfg[3] input.</i>

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the

initialization, software may enable e2eECC operation by writing the appropriate values to the E2ECTLO_{[I,D]ECCEN} control bits.

The E2ECTLO_{[I,D]DAC} control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

15.10.1.2 End-to-End ECC error generation by software

Error generation provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus (but not on cache hits).

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

Figure 127 shows the layout of E2EECSR0.

Figure 127. e2eECC Error Control/Status Register 0 (E2EECSR0)

0				RCHKBIT								0				WRC	INVC	0				WCHKBIT/CHKINVT									
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR 511; Read/Write; Reset: 0x0; Privileged

Table 222. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to p_hrchkbit[7:0] for external reads and to p_dchk[0:7] for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	—	Reserved

Table 222. E2EECSR0 field descriptions (continued)

Bits	Name	Description
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving p_hwchkbit[7:0] with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving p_hwchkbit[7:0] with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved <i>Note: Checkbit substitution has priority over Error injection.</i>
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying p_hwchkbit[7:0] based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying p_hwchkbit[7:0] based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved <i>Note: Checkbit substitution has priority over Error injection.</i>
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to p_hwchkbit[7:0]. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on p_hwchkbit[7:0] when error injection occurs.

16 System Integration Unit Lite2 (SIUL2)

16.1 Introduction

16.1.1 Overview

The System Integration Unit Lite2 (SIUL2) provides control over all the I/O ports on this device and supports 13 ports with 16 bits of bidirectional, general-purpose input and output signals. It supports nine external interrupts with trigger event configuration. [Figure 128](#) is the block diagram of SIUL2 and its interfaces to other system components for this device.

The SIUL2 provides the GPIO ports on the device.

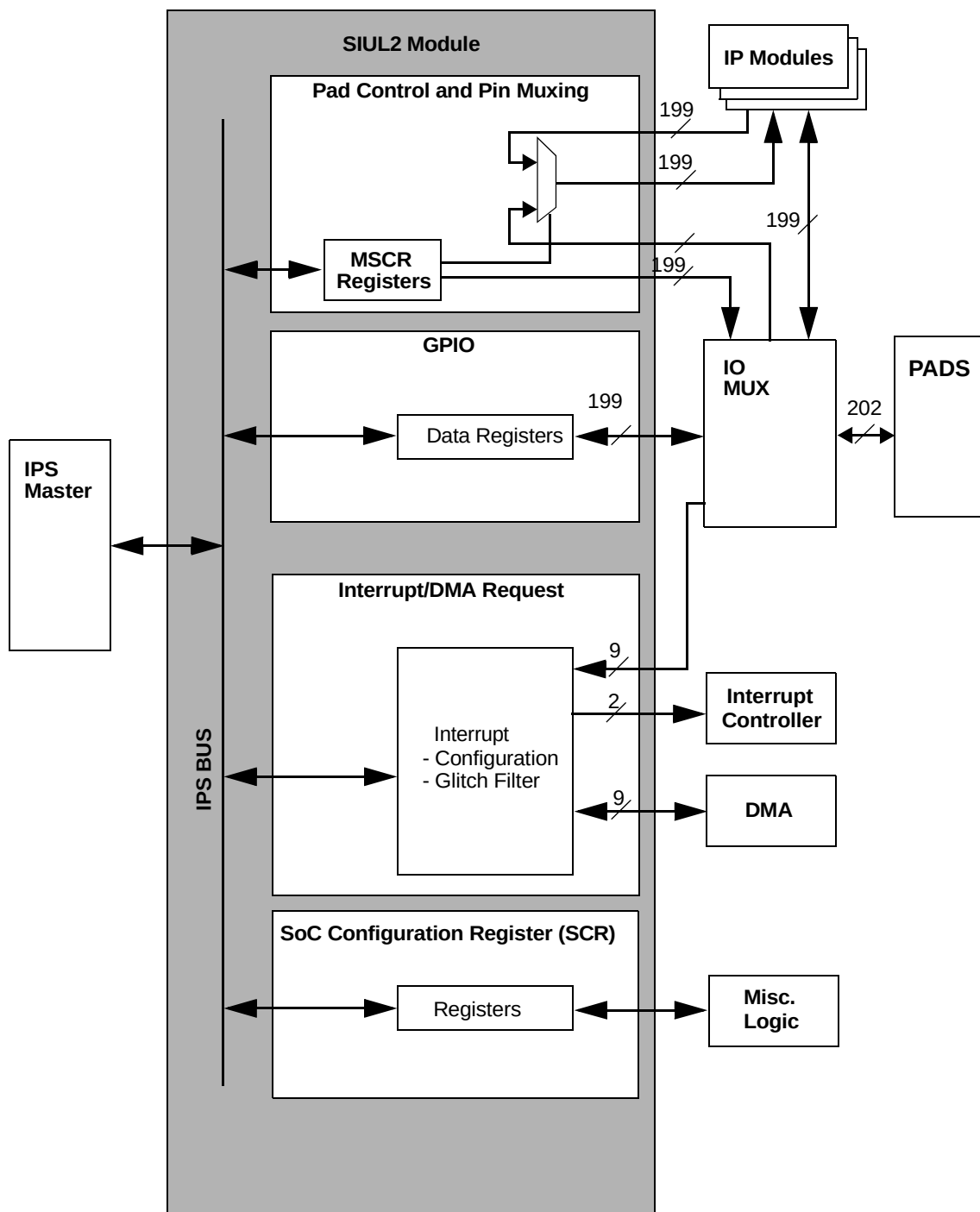
- When configured as output, you can write to the internal GPDO register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from the internal GPDI register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking the driven value on the associated I/O pad.

In order to assist in software development, there are different access mechanisms to the GPIO data registers. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation.

- Access to two 16-bit ports in one access
- Read/Write access to a single bit
- A 16-bit port write with a bit mask, using single 32-bit access

There are nine external interrupt/DMA inputs to the SIUL2 that map to EIRQ pins on the device. The interrupt/DMA sources can be configured to have a digital filter to reject short glitches on the inputs.

Figure 128. SIUL2 block diagram



16.1.2 Features

The SIUL2 supports these distinctive features:

- 13 GPIO ports with data control
 - Drive data to up to 16 independent I/O channels
 - Sample data from up to 16 independent I/O channels

Two 16-bit registers can be read/written with one access to for a 32-bit port if needed.

External interrupt/DMA request support with:

- 2 system interrupt vectors for 9 interrupt sources (9 connected to EIRQ pins for this device) with independent interrupt mask
- 9 programmable digital glitch filters (one for each EIRQ)
- Independent DMA channel for each EIRQ pin
- Edge detection

Additionally the SIUL2 contains the multiplexed signal configuration registers that configure the electrical parameters and settings of all 202 functional pads. These are used to configure the following pad features:

- Drive strength
- Output impedance control
- Open drain/source output enable
- Slew rate control of JTAG and FlexRay pins
- Internal weak pull control
- Pin function assignment
- Control of analog path switches
- Safe mode behavior configuration
- TTL, CMOS, and Automotive input levels

16.1.3 Register protection

The System Integration Unit Lite2 uses the Register Protection Scheme to protect the individual registers from accidental writes. The following registers can be protected:

- SIUL2 DMA/Interrupt Request Enable Register 0(SIUL2_DIRER0)
- SIUL2 DMA/Interrupt Request Select Register 0(SIUL2_DIRSR0)
- SIUL2 Interrupt Rising-Edge Event Enable Register 0(SIUL2_IREER0)
- SIUL2 Interrupt Falling-Edge Event Enable Register 0(SIUL2_IFEER0)
- SIUL2 Interrupt Filter Enable Register 0(SIUL2_IFER0)
- SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31)
- SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)
- SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0–SIUL2_MSCR1023)

Details about the protected registers at specific addresses can be found in the [Chapter 83: Register Protection \(REG_PROT\)](#) and [Chapter 7: Device Configuration](#).

16.2 Memory map and register description

This section provides a detailed description of all registers accessible in the SIUL2 module.

16.2.1 Memory map

[Table 223](#) gives an overview of the SIUL2 registers implemented.

Table 223. SIUL2 memory map

Address offset	Register name	Access	Size	Supported access (bit)	Reset value	Location
0x0000	Reserved					
0x0004	SIUL2 MCU ID Register #1 (SIUL2_MIDR1)	R	32	32	0x5746_0000	on page 611
0x0008	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	R	32	32	0x4000_4D00	on page 612
0x000C–0x000F	Reserved					
0x0010	SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)	R/W	32	32/16/8	0x0000_0000	on page 613
0x0014–0x0017	Reserved					
0x0018	SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)	R/W	32	32/16/8	0x0000_0000	on page 614
0x001C–0x001F	Reserved					
0x0020	SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)	R/W	32	32/16/8	0x0000_0000	on page 615
0x0024	Reserved					
0x0028	SIUL2 Interrupt Rising-Edge Event Enable Register0 (SIUL2_IREEER0)	R/W	32	32/16/8	0x0000_0000	on page 616
0x002C	Reserved					
0x0030	SIUL2 Interrupt Falling-Edge Event Enable Register0 (SIUL2_IFEER0)	R/W	32	32/16/8	0x0000_0000	on page 617
0x0034	Reserved					
0x0038	SIUL2 Interrupt Filter Enable Register0 (SIUL2_IFER0)	R/W	32	32/16/8	0x0000_0000	on page 618
0x003C	Reserved					
0x0040–0x00BC	SIUL2 Interrupt Filter Maximum Counter Register0–Interrupt Filter Maximum Counter Register31 (SIUL2_IFMCR0–SIUL2_IFMCR31)	R/W	32	32	0x0000_0000	on page 619
0x00C0	SIUL2 Interrupt Filter Clock Prescaler (SIUL2_IFCPR)	R/W	32	32	0x0000_0000	on page 620
0x00C4–0x00FF	Reserved					

Table 223. SIUL2 memory map (continued)

Address offset	Register name	Access	Size	Supported access (bit)	Reset value	Location
0x0100	SIUL2 SoC Configuration Register0 (SIUL2_SCR0)	R/W	16	16	0x0000_0000	on page 620
0x0102–0x023F	Reserved					
0x0240–0x123C	SIUL2 Multiplexed Signal Configuration Register0–Multiplexed Signal Configuration Register1023 (SIUL2_MSCR0–SIUL2_MSCR1023)	R/W	32	32	0x0000_0000	on page 621
0x1240–0x12FF	Reserved					
0x1300–0x14FF	SIUL2 GPIO Pad Data Out Register0–GPIO Pad Data Out Register511 (SIUL2_GPDO0–SIUL2_GPDO511)	R/W	8	32/16/8	0x0000_0000	on page 626
0x1500–0x16FF	SIUL2 GPIO Pad Data In Register0–GPIO Pad Data In Register511 (SIUL2_GPDI0–SIUL2_GPDI511)	R	8	32/16/8	0x0000_0000	on page 626
0x1700–0x173E	SIUL2 Parallel GPIO Pad Data Out Register0–Parallel GPIO Pad Data Out Register31 (SIUL2_PGPDO0–SIUL2_PGPDO31)	R/W	16	32/16/8	0x0000_0000	on page 627
0x1740–0x177E	SIUL2 Parallel GPIO Pad Data In Register0–Parallel GPIO Pad Data In Register31 (SIUL2_PGPDIO0–SIUL2_PGPDIO31)	R	16	32/16/8	0x0000_0000	on page 628
0x1780–0x17FC	SIUL2 Masked Parallel GPIO Pad Data Out Register0–Masked Parallel GPIO Pad Data Out Register31 (SIUL2_MPGPDO0–SIUL2_MPGPDO31)	R/W	32	32	0x0000_0000	on page 628

Note: Reserved registers will be read as 0, write will have no effect. If supported and enabled in this MCU, a transfer error will be issued when trying to access completely reserved register space. 8/16 for when defined as only 32-bit access gives transfer error. Unimplemented registers accesses give transfer error i.e. due to register not being present.

16.2.2 Register descriptions

This section describes in address order all the SIUL2 registers. Each description includes a standard register diagram with an associated figure number. .

16.2.2.1 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

This register contains identification information about the device.

Figure 129. SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

Address: 0x0004

Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	0	1	0	1	0	1	1	1	0	1	0	0	0	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ED	PKG				0	0		MAJOR_MASK				MINOR_MASK			
W																
Reset	0	0 ⁽¹⁾	0 ⁽¹⁾	0 ⁽¹⁾	0 ⁽¹⁾	0 ⁽¹⁾	0	0	0 ⁽²⁾	0 ⁽²⁾	0 ⁽²⁾	0 ⁽²⁾	0 ⁽²⁾	0 ⁽²⁾	0 ⁽²⁾	0 ⁽²⁾

1. Values are set at factory and cannot be modified.
2. Value is set at factory and will vary each revision of device.

Table 224. SIUL2_MIDR1 field descriptions

Field	Description
0:15 PARTNUM	MCU Part Number—These digits identify the part number of the chip. Default is 5746.
16 ED	Always reads back 0.
17:21 PKG	Package Settings—Can be read by software to determine the package type that is used for the particular device: 0b10001: 176-pin QFP 0b10011: 292-pin BGA All other bit combinations are reserved for future use.
22:23	Reserved
24:27 MAJOR_MASK	Major Mask Revision.
28:31 MINOR_MASK	Minor Mask Revision.

16.2.2.2 SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

This register contains identification information about the device.

Figure 130. SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

Address: 0x0008

Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE_1				FLASH_SIZE_2				0	0	0	0	0	0	0
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FAMILYNUM								0	0	0	0	0	0	0	0
W																
Reset	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0

Table 225. SIUL2_MIDR2 field description

Field	Description
0 SF	Manufacturer: 0 Freescale 1 STMicroelectronics
1:4 FLASH_SIZE_1	Coarse granularity for Flash memory size. Needs to be combined with FLASH_SIZE_2 to calculate the actual memory size: 0b000016K 0b000132K 0b001064K 0b0011128K 0b0100256K 0b0101512K 0b01101024K 0b01112048K 0b10004096K 0b10018192K 0b101016384K 0b1110256M 0b1111512M

Table 225. SIUL2_MIDR2 field description (continued)

Field	Description
5:8 FLASH_SIZE_2	Fine granularity for Flash memory size. Needs to be combined with FLASH_SIZE_1 to calculate the actual memory size: 0b00000 x (FLASH_SIZE_1 / 8) 0b00011 x (FLASH_SIZE_1 / 8) 0b00102 x (FLASH_SIZE_1 / 8) 0b111014 x (FLASH_SIZE_1 / 8) 0b111115 x (FLASH_SIZE_1 / 8)
9:15	Reserved
16:23 FAMILYNUM	ASCII character in MCU Part Number: 0x4DM All other values reserved for future use.
24:31	Reserved

16.2.2.3 SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)

The DMA/Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREEER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set irrespective of the corresponding DMA/Interrupt Request Enable bit in DMA/Interrupt Request Enable Register (SIUL2_DIRER0) is enabled. The EIF bit remains set until cleared by software or through the servicing of a DMA request. The EIF bits are cleared by writing a '1' to the bits. A write of '0' has no effect.

This register contains the interrupt flags.

Figure 131. SIUL2 DMA/Interrupt Status Flag Register (SIUL2_DISR0)

Address: 0x0010

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	EIF11	EIF10	EIF9	EIF8	0	0	EIF5	EIF4	0	EIF2	EIF1	EIF0
W					w1c	w1c	w1c	w1c			w1c	w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 226. SIUL2_DISR0 field descriptions

Field	Description
0:20	Reserved
21:23 EIF11 EIF10 EIF9 EIF8	External Interrupt Status Flag x—This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
24:25	Reserved
26:27 EIF5 EIF4	External Interrupt Status Flag x—This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
28	Reserved
29:31 EIF2 EIF0	External Interrupt Status Flag x—This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

16.2.2.4 SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)

The DMA/Interrupt Request Enable Register enables the assertion of DMA or interrupt request if the corresponding External IRQ Flag bit is set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0). This register enables the interrupt messaging to the interrupt controller.

Figure 132. SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

Address: 0x0018

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	EIRE 11	EIRE 10	EIRE 9	EIRE 8	0	0	EIRE 5	EIRE 4	0	EIRE 2	EIRE 1	EIRE 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 227. SIUL2_DIRER0 field descriptions

Field	Description
0:20	Reserved
21:23 EIRE11 EIRE10 EIRE9 EIRE8	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
24:25	Reserved
26:27 EIRE5 EIRE4	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
28	Reserved
29:31 EIRE2 EIRE0	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR

16.2.2.5 SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)

The DIRSR selects between the DMA or interrupt request. If the corresponding bits are set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0) and SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0), then the DMA/Interrupt Request Select bit determines whether DMA or an interrupt request is asserted. EIRQ are the external interrupt package pins on the device.

Figure 133. SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

Address: 0x0020

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	DIRS 11	DIRS 10	DIRS 9	DIRS 8	0	0	DIRS 5	DIRS 4	0	DIRS 2	DIRS 1	DIRS 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 228. SIUL2_DIRSR0 field descriptions

Field	Description
0:20	Reserved
21:23 DIRS11 DIRS10 DIRS9 DIRS8	DMA/Interrupt Request Select Register—Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
24:25	Reserved
26:27 DIRS5 DIRS4	DMA/Interrupt Request Select Register—Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
28	Reserved
29:31 DIRS2 DIRS0	DMA/Interrupt Request Select Register—Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected

16.2.2.6 SIUL2 Interrupt Rising-Edge Event Enable Register0 (SIUL2_IREE0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

Figure 134. SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREE0)

Address: 0x0028

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	IREE 11	IREE 10	IREE 9	IREE 8	0	0	IREE 5	IREE 4	0	IREE 2	IREE 1	IREE 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 229. SIUL2_IREE0 field descriptions

Field	Description
0:20	Reserved
21:23 IREE11 IREE10 IREE9 IREE8	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
24:25	Reserved
26:27 IREE5 IREE4	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
28	Reserved
29:31 IREE2 IREE0	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

16.2.2.7 SIUL2 Interrupt Falling-Edge Event Enable Register0 (SIUL2_IFEER0)

This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

Figure 135. SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

Address: 0x0030

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	IFEE	IFEE	IFEE	IFEE	0	0	IFEE	IFEE	0	IFEE	IFEE	IFEE
W					11	10	9	8			5	4		2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 230. SIUL2_IFEER0 field descriptions

Field	Description
0:20	Reserved
21:23 IFEE11 IFEE10 IFEE9 IFEE9	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
24:25	Reserved
26:27 IFEE5 IFEE4	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
28	Reserved
29:31 IFEE2 IFEE0	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

Note: If both IREE bit and IFEE bit are cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

16.2.2.8 SIUL2 Interrupt Filter Enable Register0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs.

Figure 136. SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

Address: 0x0038

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	IFE 10	IFE 10	IFE9	IFE8	0	0	IFE5	IFE4	0	IFE2	IFE1	IFE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 231. SIUL2_IFER0 field descriptions

Field	Description
0:20	Reserved
21:23 IFE11 IFE10 IFE9 IFE8	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
24:25	Reserved
26:27 IFE5 IFE4	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
28	Reserved
29:31 IFE2 IFE0	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

16.2.2.9 SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0 – SIUL2_IFMCR31)

These registers are used to configure the filter counter associated with each digital glitch filter.

Figure 137. SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0)

Address: 0x0040 – 0x00BF

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MAXCNT0 [3:0]			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 232. SIUL2_IFMCR0 field descriptions

Field	Description
28:31 MAXCNTx	Maximum Interrupt Filter Counter setting Filter Period = TCK*MAXCNTx + n*TCK where (n can 0 to 4) MAXCNTx can be 0 to 15 (for MAXCNT<3 filter will behave as all PASS filter) TCK Prescaled Filter Clock Period, which is IRC clock prescaled to IFCPR value specified in SIUL2_IFCPR T(IRC) Basic Filter Clock Period: 62.5 ns (F=16 MHz) Also note that Filter delay is 2 TCK clock cycles more than Filter period.

16.2.2.10 SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

This register is used to configure a clock prescaler which is used to select the clock for all digital filters. Prescaler is applied to the input clock to the SIUL2, which is the system PBRIDGE clock counters in the SIUL2. See [Chapter 27: Clocking](#) for information on clocking.

Figure 138. SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

Address: 0x00C0																Access: User read/write																		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IFCP[3:0]					
W																																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 233. SIUL2_IFCPR field descriptions

Field	Description
28:31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15

16.2.2.11 SoC Configuration Register0 (SIUL2_SCR0)

The FEC_MODE bit is used to select between the Media Independent Interface (MII) and the Reduced Media Independent Interface (RMII) mode for the Fast Ethernet Controller (FEC)

Figure 139. SIUL2 SoC Configuration Register0 (SIUL2_SCR0)

Address:0x0100												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FEC_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	MODE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 234. SIUL2_SCR0 field descriptions

Field	Description
0 FEC_MODE	FEC mode setting. 0 RMII is selected. 1 MII is selected. <i>Note: MII/RMII pad configuration is done independently in the SIUL2 MSCR registers.</i>

16.2.2.12 SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0–SIUL2_MSCR1023)

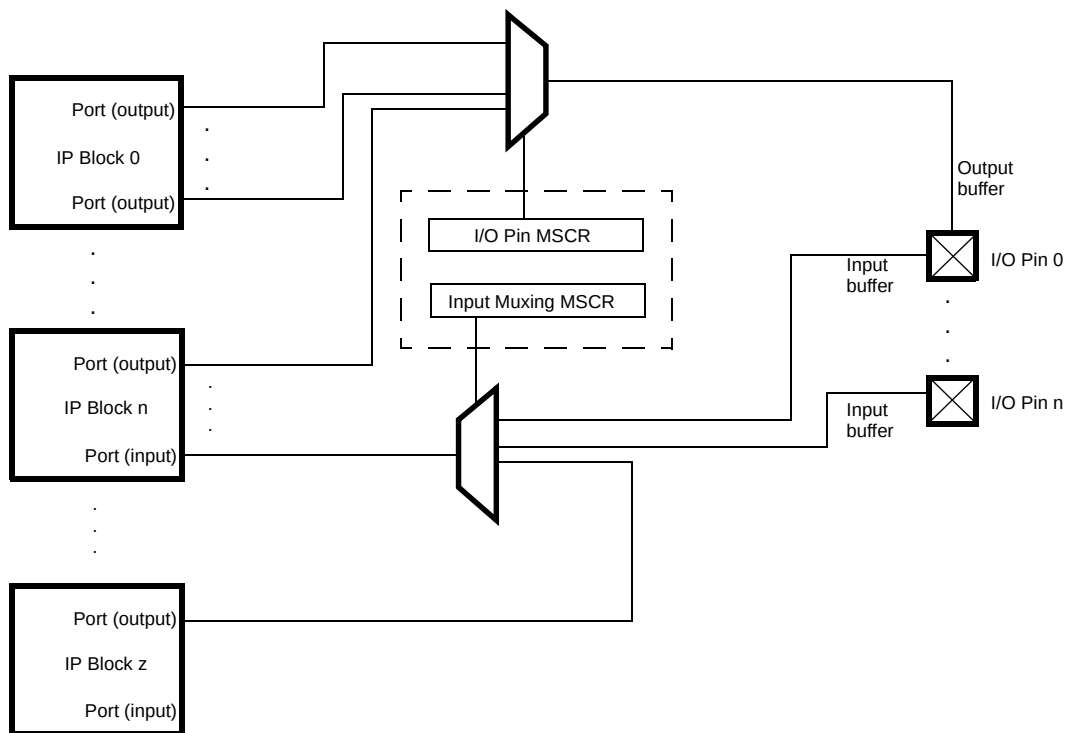
The MSCR registers control the I/O pin function and electrical properties for each pin on the device, and are used to select the input signal for IP blocks on the device that have multiple input sources. There is a dedicated MSCR register for each I/O pin on the device. Each MSCR register is independent of the others, except in the case of an LVDS configuration, where two MSCR registers must be configured together in this mode.

The MSCR registers are divided into two sets:

- I/O pin control registers, see [Section 16.2.2.13, I/O Pin Multiplexed Signal Configuration Registers \(SIUL2_MSCR0 - SIUL2_MSCR511\)](#).
- Multiplexed input selection registers [Section 16.2.2.14, Multiplexed Signal Configuration Registers for Multiplexed Input Selection \(SIUL2_MSCR512 - SIUL2_MSCR1023\)](#).

The signal data flow from IP blocks on the device to and from the I/O pins, and from IP block to IP block is given in [Figure 140](#).

Figure 140. MSCR I/O Pin and IP Block Port Connectivity



16.2.2.13 I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0 - SIUL2_MSCR511)

Note:

The I/O pin MSCR register definition is given in [Figure 141](#). These registers select the output function on the associated I/O pin. The I/O MSCR registers control the drive strength, output drive circuit, pull up/down, input buffer enable, input hysteresis, input level selection, safe-mode operation, and analog input enable of the associated I/O pin. *Not all pads have all bits i.e., only some pads have hysteresis. See the SPC57EM80xx I/O Signal Description*

and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

When an I/O pin is configured as a single-ended input, the input buffer must be enabled for the pin using the MSCR[IBE] bit. This enables the input to all input destinations connected to the pin, including the GPIO input. In this way, the GPIO can be read at any time, regardless of the pin function. For IP blocks that have multiple input sources, the input source is selected in the multiplexed input selection MSCRs, and the associated I/O pin MSCR for the selected input must have the MSCR[IBE] bit enabled.

Figure 141. I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0 - SIUL2_MSCR511)

Address: 0x0240–0xA3C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	OERC			0	ODC			SMC	APC	ILS		IBE	HYS	WPD E	WPU E
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV								SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 235. SIUL2_MSCR0 - SIUL2_MSCR511 field description

Field	Description
0	Reserved
1:3 OERC	Output Edge Rate Control—Specifies the output impedance (OI) of the associated pin. 000 Weak drive (800 ohm) 001 Medium drive (200 ohm) 010 Strong drive (50 ohm/10 ns) 011 Very strong drive (50 ohm/5 ns) 100 6 pF 101 12 pF 110 18 pF 111 30 pF Refer to output Impedance columns in the I/O Signal Description Table for the applicable edge rates for each I/O pin on the device. The SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) are attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. See the device data sheet for the electrical characteristics of the weak, medium, strong, and very strong pad types. <i>Note: OERC[2] programming is required for forward compatibility.</i>
4	Reserved

Table 235. SIUL2_MSCR0 - SIUL2_MSCR511 field description (continued)

Field	Description
5:7 ODC	Output Drive Control—Specifies the type of output drive control for the associated pin. 000 Output buffer disabled 001 Open-drain 010 Push-pull 011 Open-source 100 Microsecond Channel LVDS 101 LFAST LVDS 110 - 111 Reserved
8 SMC	Safe Mode Control—Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode . 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Don't disable
9 APC	Analog Pad Control—Enables the pin's analog-input-path switch if the associated pin has an analog input function. 0 Disable (the switch is off) 1 Enable (another module can control the state of the switch)
10:11 ILS	Input Level Selection—Specifies the logic family for the associated pin, which determines its logic switching levels. 00 Automotive 01 TTL 10 LVDS 11 CMOS See the device data sheet for the electrical characteristics of the I/O pad input buffer types.
12 IBE	Input Buffer Enable—Enables the associated pin's input buffer. 0 Disabled 1 Enabled
13 HYS	Input Hysteresis—Enables input hysteresis for the associated pin. 0 Disabled 1 Enabled
14 WPDE	Weak Pulldown Enable—Used only when the associated destination is a chip pin. Enables the associated pin's weak pulldown resistor. It is OK for both WPDE and WPUE to be enabled, if they are, the pad is neither a weak pullup and a weak pulldown. 0 Disabled 1 Enabled
15 WPUE	Weak Pullup Enable—Used only when the associated destination is a chip pin. Enables the associated pin's weak pullup resistor. It is OK for both WPDE and WPUE to be enabled, if they are, the pad is neither a weak pullup and a weak pulldown. 0 Disabled 1 Enabled
16 INV	Invert—The output selected by the corresponding MSCR SSS field can be inverted before it is driven on the I/O pin with the INV bit. 0 The output selected by the SSS field is not inverted before being driven to the pin 1 The output selected by the SSS field is inverted before being driven to the pin <i>Note: Use of the INV bit is not supported when the ODC is configured for open-drain or open-source modes, and the output data will not be as expected for an inverted open-drain or open-source connection.</i>

Table 235. SIUL2_MSCR0 - SIUL2_MSCR511 field description (continued)

Field	Description
17:23	Reserved
24:31 SSS	Source Signal Select—Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. See Section 16.2.2.15, “Chip-pin MSCR assignments and related information” and Section 16.2.2.16, Module-port MSCR assignments and SSS values.”

All bits and fields in the I/O MSCR registers are applicable for all GPIO ports on the device. This includes all ports on the device with digital output function. For ADC input ports with digital input only (no output), the OERC, ODC, SMC, INV, and SSS bits/fields do not apply. The SSS is always 8b0 for input only ports, enabling the general purpose input. The analog input path is enabled in the ADC logic when the channel is selected for conversion.

The MSCR reset state given in [Figure 141](#) applies to all GPIO pins on the devices, which defaults to input/pull-up enabled. The exceptions to this are documented in [Table 236](#). Only the SMC, HYS, WPDE, and WPUE bits have exceptions to the default MSRC reset value.

Table 236. I/O MSCR Reset State Exceptions

Port	Critical Function	SMC	HYS	WPDE	WPUE
PA[5]	JCOMP	1	1	1	0
PA[6]	TCK	1	1	1	0
PA[7]	TMS	1	1	1	1
PA[8]	TDI	1	1	0	1
PA[9]	TDO	1	1	0	1 ⁽¹⁾
PA[14]	-	1	1	0	1
PB[11]	ERROR	1	0	0	0
PC[10:15]	-	1	1	0	1
PD[6:7]	-	1	1	0	1
PE[12]	-	1	1	0	1
PF[13]	-	1	1	0	1
PI[14:15]	-	1	1	0	1
PK[12:15]	-	1	1	0	1
PM[0:8]	-	1	1	0	1

1. When JTAG is enabled (JCOMP asserted) during reset, the TDO pin is pulled up during reset, but hi-z after. If JTAG is disabled, the TDO pin defaults to input/pull-up during and after reset.

16.2.2.14 Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512 - SIUL2_MSCR1023)

The multiplexed input selection MSCR register definition is given in [Figure 142](#). These registers select the input source for IP blocks on the device that have inputs from multiple

sources, both other IP blocks and I/O pins. It is possible to configure the pad to be in input without having to connect it externally. This is possible, either by enabling internal pull-up/pull-down or disabling the input buffer (but in this case data is seen as 0 internally).

Figure 142. Multiplexed Signal Configuration Registers for Multiplexed Input Selection

Address: 0x0A40–0x123C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV								SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(SIUL2_MSCR512 - SIUL2_MSCR1023)

Table 237. SIUL2_MSCR512 - SIUL2_MSCR1023 field description

Field	Description
0:15	Reserved
16 INV	Invert— The input source signal selected by the corresponding SSS field for this MSCR can be inverted before it connects to the destination input with the INV bit. 0 SSS selected input signal is connected directly to the destination input 1 SSS selected input signal is inverted before it is connected to the destination input
17:23	Reserved
24:31 SSS	Source Signal Select—Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. See Section 16.2.2.15, “Chip-pin MSCR assignments and related information” and Section 16.2.2.16, Module-port MSCR assignments and SSS values .

16.2.2.15 Chip-pin MSCR assignments and related information

For device pin MSCR register definitions, see the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

16.2.2.16 Module-port MSCR assignments and SSS values

For each internal module port that is or can be configured as an input and has more than one possible source. See the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the Input

Multiplexing Table tab. The Input Multiplexing Table shows the assigned MSCR, and the SSS values and their corresponding source signals.

16.2.2.17 SIUL2 GPIO Pad Data Output Registers (SIUL2_GPDO0_3–SIUL2_GPDO508_511)

These registers can be used to set or clear a single GPIO pad with a byte access. For mapping of ports to GPDO registers, see the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. Note that the GPDO registers do not apply for ports PA[5], PA[6], PA[7], and PL[8]. The PA ports are JTAG pins with no GPIO, and PL[8] is not included on the device. Writes to the GPDO register corresponding to these ports have no effect.

Figure 143. SIUL2 GPIO Pad Data Output register 0_3 (SIUL2_GPDO0_3)

Address: 0x1300-0x14FF

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	PDO [0]	0	0	0	0	0	0	0	PDO [1]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	PDO [2]	0	0	0	0	0	0	0	PDO [3]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 238. SIUL2_GPDO0_3 field description

Field	Description
PDO[x]	Pad Data Out—This bit stores the data to be driven out on the external GPIO pad controlled by this register. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

16.2.2.18 SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDIO_3 – SIUL2_GPDIO508_511)

These registers can be used to read the GPIO pad data with a byte access. For mapping of ports to GPDIO registers, See the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. Note that the GPDIO registers do not apply for ports PA[5], PA[6], PA[7], and PL[8]. The PA ports are JTAG pins with no GPIO, and PL[8] is not included on the device. The GPDIO values corresponding to these ports always read a value of zero.

Figure 144. SIUL2 GPIO Pad Data Input register 0_3 (SIUL2_GPDI0_3)

Address: 0x1500-0x16FF

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	PDI[0]	0	0	0	0	0	0	0	PDI[1]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	PDI[2]	0	0	0	0	0	0	0	PDI[3]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 239. SIUL2_GPDI0_3 field description

Field	Description
PDI[x]	Pad Data In—This bit stores the value of the external GPIO pad associated with this register. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

16.2.2.19 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0 – SIUL2_PGPDO31)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write. Note that the PGPDO registers do not apply for ports PA[5], PA[6], PA[7], and PL[8]. The PA ports are JTAG pins with no GPIO, and PL[8] is not included on the device. Writes to the PGPDO register corresponding to these ports have no effect.

Figure 145. SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0)

Address: 0x1700-0x173E

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PPDO[x][0:15]																PPDO[x+1][0:15]															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 240. SIUL2_PGPDO0 field description

Field	Description
0:15 PPDO[x]	Parallel Pad Data Out—Write or read the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO). The x and y bit index define which PPDO register bit is equivalent to which PDO register bit according to the following equation: $PPDO[x][y] = PDO[(x*16)+y]$

Note: The SIUL2_PGPDO registers access the same physical resource as the SIUL2_PDO and SIUL2_MPGPDO address locations. Some examples of the mapping:

$PPDO[0][0] = PDO[0]$

$PPDO[2][0] = PDO[32]$

$PPDO[31][15] = PDO[511]$

16.2.2.20 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0 – SIUL2_PGPDI31)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read. Note that the GPDI registers do not apply for ports PA[5], PA[6], PA[7], and PL[8]. The PA ports are JTAG pins with no GPIO, and PL[8] is not included on the device. The GPDI values corresponding to these ports always read a value of zero.

Figure 146. SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0)

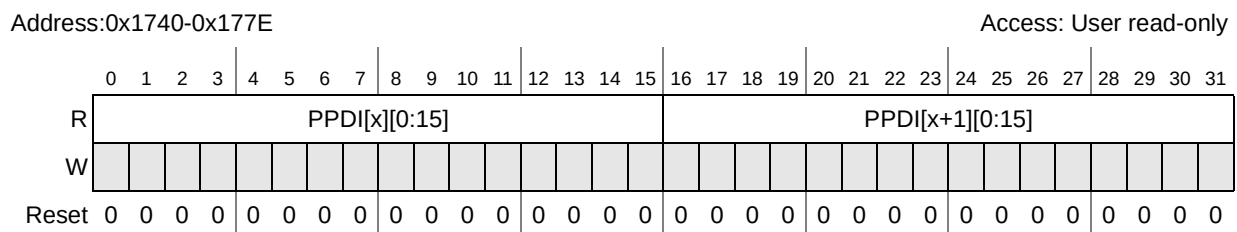


Table 241. SIUL2_PGPDI0 field description

Field	Description
0:15 PPDI[x]	Parallel Pad Data In —Reads the current pad value. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and y bit index define which PPDI register bit is equivalent to which PDI register bit according to the following equation: $PPDI[x][y] = PDI[(x*16)+y]$

16.2.2.21 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0–SIUL2_MPGPDO31)

This register can be used to selectively modify the pad values associated to PPDO[x][0:15]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read

access will return 0. Note that the MPGPDO registers do not apply for ports PA[5], PA[6], PA[7], and PL[8]. The PA ports are JTAG pins with no GPIO, and PL[8] is not included on the device. Writes to the MPGPDO register corresponding to these ports have no effect.

Figure 147. SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0)

Address: 0x1780-0x17FC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W	MASK[x][0:15]															MPPDO[x][0:15]																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 242. SIUL2_MPGPDO0 field description

Field	Description
MASK[x] [0:15]	Mask Field—Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0 The associated bit value in the MPPDO[x] field is ignored 1 The associated bit value in the MPPDO[x] field is written
MPPDO[x] [0:15]	Masked Parallel Pad Data Out—Write the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO). The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: $MPPDO[x][y] = PDO[(x*16)+y]$

16.3 Functional description

16.3.1 General

This section provides a complete functional description of the SIUL2.

16.3.2 Pad control

The SIUL2 controls the electrical characteristics of 199 pads on the device through the MSCR registers. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2.

The required controls are:

- Slew rate control
 - This is needed to offer improved EMC performance.
 - Support for a fast and a slow slew rate.
- Output Impedance Control
 - 4 output impedances supported
- Internal weak pull capability
 - This is needed to offer flexibility in the device configuration and the elimination of external hardware in some cases.
 - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down or no-pull enabled.
- Control of analog path switches
- Safe Mode behavior configuration
- Open drain/source enable
 - Needed to support different pads muxed (e.g. mux IIC, with non open drain pads)
- Pin Function Assignment
 - This setting defines which chip function has control over the output of the pad.

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for the user software to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register and allowing simplified duplication of software for each pad with indexed changes for each pad.

16.3.3 General purpose input or output pads (GPIO)

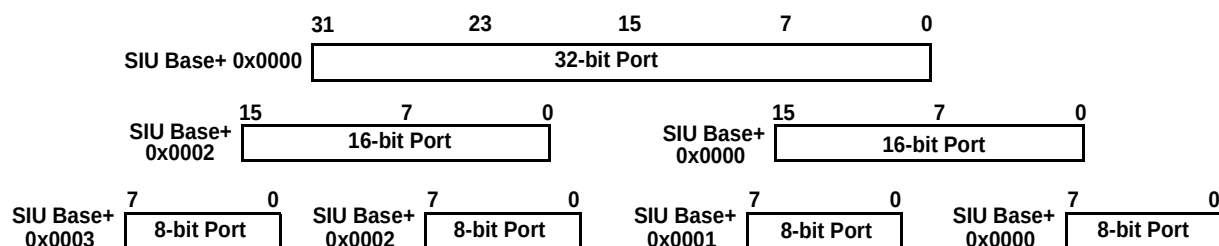
The SIUL2 allows each pad to be configured as either a General Purpose Input Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage 199 GPIO pads organized as ports that can be accessed for data reads and writes as 32-bit, 16-bit or 8-bit.

As shown in [Figure 148](#), all port accesses are identical with each read or write being performed only at a different location to access a different port width.

Figure 148. Data Port example arrangement showing configuration for different port width accesses



This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

All general purpose pads are implemented as bidirectional.

Note: In case the bi-directional operation impacts some performances (e.g. ADC accuracy), or is not required for a specific pad function it is acceptable to limit the functionality of some pads to "Input Only".

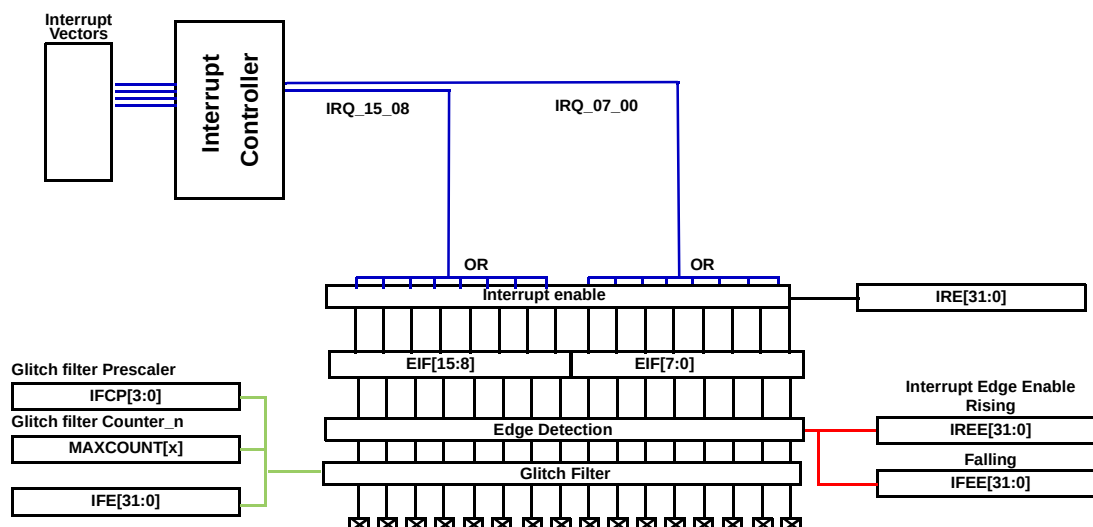
16.3.4 External interrupts/DMA requests (EIRQ pins)

The SIUL2 supports nine external interrupts which are allocated to EIRQ pins on the device.

The SIUL2 supports two interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support eight external interrupt sources from the device pads.

Refer to [Figure 149](#) for an overview of the external interrupt implementation.

Figure 149. External interrupt pad diagram



All the external interrupt pads within a single group (max 8 pads) have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

An MCU specific number of external interrupt lines can have a digital glitch filters applied to them. The supported range is 1 to 8. The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on an interrupt line.

16.3.4.1 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request, during pin interrupt initialization, the user must do the following:

- Mask interrupts by clearing the EIREn bits in DIRER0.
- Select the pin polarity by setting the appropriate IREEN bits in IREER0 and the appropriate IFEEEn bits in IFEER0 as desired.
- Configure the appropriate bits in the MSCR[0–511] register for the external interrupt pin(s) desired as follows:
 - Clear the ODC bits to do disable output.
 - Set the IBE bit to enable the pin's input buffer.
 - If using the internal weak pullup/pulldown, configure the appropriate WPUE and WPDE bits.

Note: MSCR_SSS bits do not need to be changed for either MSCR[0:511] or MSCR[512:1023] since the external interrupt pin input is directly connected to the SIUL for EIRQ pins.

External interrupt pins should never be configured as outputs (MSCR_ODC bits are not zeros) when external interrupt inputs are desired since false interrupts could be detected (such as from a GPIO configuration).

- Select the request desired between DMA or Interrupt by writing the appropriate DIRSn bits in DIRSR0.
- Select the desired glitch filter setup for the pins by writing the following:
 - Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMCn register for the respective external interrupt that is being used.
 - Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR register.
 - Then enable the glitch filter for the desired external interrupt pins by setting the appropriate IFEn bits in IFER0.

Note: The glitch filter clock runs on the internal RC oscillator clock whereas the SIUL logic runs off of the PBRIDGEEx clock. The IRC clock is at a fixed frequency of 16 MHz. The PBRIDGEEx clock can run as high as 50 MHz when core clock is at 200 MHz.

The glitch filter is enabled on IRC clock, so make sure at least one IRC clock occurs after enabling it before trying to detect glitchless events.

- Write to EIFn bits in DISR0 to as desired to clear any flags (For DMA, its self clearing).
- Enable the interrupt pins by setting the appropriate EIREn bits in DIREER0.

16.3.4.2 External interrupt management

Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (SIUL2_DIRER0 - [Figure 132](#)). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled is reserved and should not be configured.

The active IRQ edge is controlled by the users through the configuration of the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (DISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

The SPC57EM80xx device supports two SIUL2 interrupt vectors aggregating a total of eight external interrupt sources as shown in [Figure 150](#).

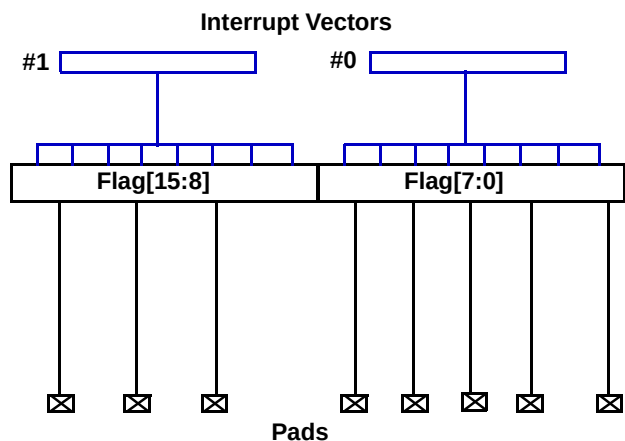


Figure 150. Interrupt to vector mapping at MCU level

16.3.4.3 External interrupt request

The EIRQ input pins on the device are sources for interrupt or DMA requests. The nine EIRQ pins for the device map to two interrupt requests. The mapping of interrupt requests and EIRQ pins is given in [Table 243](#).

Table 243. Interrupt source mapping to SIUL2 interrupt request output

Interrupt requests	EIRQ pins
0	EIRQ0 EIRQ1 EIRQ2 EIRQ4 EIRQ5
1	EIRQ8 EIRQ9 EIRQ10 EIRQ11

16.3.4.4 Interrupt Vector

The nine EIRQ pins on the device map to an Interrupt Vector in the Interrupt Controller.

Note: See the [Chapter 7: Device Configuration](#) chapter, [Table 54](#) for Interrupt Vector table.

16.3.4.5 DMA requests

The nine EIRQ pins on the device map to an independent DMA request channel in the DMA controller.

Note: See the [Chapter 7: Device Configuration](#), [Table 57](#) for DMA request mapping table.

17 Crossbar Switch (XBAR)

17.1 Introduction

This chapter provides information on the layout, configuration, and programming of the crossbar switch. The crossbar switch connects bus masters and bus slaves using a hardware interconnect matrix. This structure allows all bus masters to access different bus slaves simultaneously with no interference while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

17.1.1 Features

The crossbar switch includes these distinctive features:

- Symmetric crossbar bus switch implementation
 - Concurrent accesses from different masters to different slaves
 - Configurable slave arbitration attributes on a slave-by-slave basis
- 64-bit datapath width
- Support for 8-, 16-, 32-, and 64-bit single transfers
- Support for a variety of 4-, 8-, and 16-beat burst transfers including a 4-beat, 64-bit burst for cache line accesses
- Operation at one-to-one clock frequency with the bus masters
- Support for low-power park mode

17.1.2 Memory map and register definition

Each slave port of the crossbar switch contains configuration registers. Read and write transfers of the configuration registers require two bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The slave registers also feature a bit that, when set, prevents the registers from being written. The registers remain readable, but future write attempts have no effect on the registers and are terminated with a bus error response to the master initiating the write. The core, for example, takes a data storage interrupt.

Note: This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the Chip Configuration details for the exact master/slave assignments for your device. Additionally, all references to the crossbar switch registers are based on the physical port connections, not the logical port numbers.

Table 244. XBARx memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC00_4000	XBAR0 Priority Register Slave 0 (XBAR0_PRS0)	32	R	See Chip Config	on page 637
FC00_4010	XBAR0 Control Register 0 (XBAR0_CRS0)	32	R/W	0000_0000h	on page 640
FC00_4100	XBAR0 Priority Register Slave 1 (XBAR0_PRS1)	32	R	See Chip Config	on page 637
FC00_4110	XBAR0 Control Register 1 (XBAR0_CRS1)	32	R/W	0000_0000h	on page 640
FC00_4200	XBAR0 Priority Register Slave 2 (XBAR0_PRS2)	32	R	See Chip Config	on page 637
FC00_4210	XBAR0 Control Register 2 (XBAR0_CRS2)	32	R/W	0000_0000h	on page 640
FC00_4300	XBAR0 Priority Register Slave 3 (XBAR0_PRS3)	32	R	See Chip Config	on page 637
FC00_4310	XBAR0 Control Register 3 (XBAR0_CRS3)	32	R/W	0000_0000h	on page 640
FC00_4400	XBAR0 Priority Register Slave 4 (XBAR0_PRS4)	32	R	See Chip Config	on page 637
FC00_4410	XBAR0 Control Register 4 (XBAR0_CRS4)	32	R/W	0000_0000h	on page 640
FC00_4500	XBAR0 Priority Register Slave 5 (XBAR0_PRS5)	32	R	See Chip Config	on page 637
FC00_4510	XBAR0 Control Register 5 (XBAR0_CRS5)	32	R/W	0000_0000h	on page 640
FC00_4600	XBAR0 Priority Register Slave 6 (XBAR0_PRS6)	32	R	See Chip Config	on page 637
FC00_4610	XBAR0 Control Register 6 (XBAR0_CRS6)	32	R/W	0000_0000h	on page 640
FC00_4700	XBAR0 Priority Register Slave 7 (XBAR0_PRS7)	32	R	See Chip Config	on page 637
FC00_4710	XBAR0 Control Register 7 (XBAR0_CRS7)	32	R/W	0000_0000h	on page 640
FC00_8000	XBAR1 Priority Register Slave 0 (XBAR1_PRS0)	32	R	See Chip Config	on page 637
FC00_8010	XBAR1 Control Register 0 (XBAR1_CRS0)	32	R/W	0000_0000h	on page 640
FC00_8100	XBAR1 Priority Register Slave 1 (XBAR1_PRS1)	32	R	See Chip Config	on page 637
FC00_8110	XBAR1 Control Register 1 (XBAR1_CRS1)	32	R/W	0000_0000h	on page 640
FC00_8200	XBAR1 Priority Register Slave 2 (XBAR1_PRS2)	32	R	See Chip Config	on page 637
FC00_8210	XBAR1 Control Register 2 (XBAR1_CRS2)	32	R/W	0000_0000h	on page 640
FC00_8300	XBAR1 Priority Register Slave 3 (XBAR1_PRS3)	32	R	See Chip Config	on page 637
FC00_8310	XBAR1 Control Register 3 (XBAR1_CRS3)	32	R/W	0000_0000h	on page 640
FC00_8400	XBAR1 Priority Register Slave 4 (XBAR1_PRS4)	32	R	See Chip Config	on page 637
FC00_8410	XBAR1 Control Register 4 (XBAR1_CRS4)	32	R/W	0000_0000h	on page 640

Table 244. XBARx memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC00_8500	XBAR1 Priority Register Slave 5 (XBAR1_PRS5)	32	R	See Chip Config	on page 637
FC00_8510	XBAR1 Control Register 5 (XBAR1_CRS5)	32	R/W	0000_0000h	on page 640
FC00_8600	XBAR1 Priority Register Slave 6 (XBAR1_PRS6)	32	R	See Chip Config	on page 637
FC00_8610	XBAR1 Control Register 6 (XBAR1_CRS6)	32	R/W	0000_0000h	on page 640
FC00_8700	XBAR1 Priority Register Slave 7 (XBAR1_PRS7)	32	R	See Chip Config	on page 637
FC00_8710	XBAR1 Control Register 7 (XBAR1_CRS7)	32	R/W	0000_0000h	on page 640

17.1.2.1 XBAR Priority Registers Slave (XBARx_PRSn)

The priority registers (XBARx_PRSn) set the priority of each master port on a per-slave-port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. The XBARx_PRSn register is read-only; attempts to write to it have no effect on it and result in a bus-error response to the master initiating the write.

See [Section 7.3.3, Crossbar switch configuration](#) for XBARx_PRSn priority values.

Figure 151. Priority Registers Slave (XBARx_PRSn)

FC00_4000h (XBAR0_PRS0)		FC00_8000h (XBAR1_PRS0)	
FC00_4100h (XBAR0_PRS1)		FC00_8100h (XBAR1_PRS1)	
FC00_4200h (XBAR0_PRS2)		FC00_8200h (XBAR1_PRS2)	
FC00_4300h (XBAR0_PRS3)		FC00_8300h (XBAR1_PRS3)	
FC00_4400h (XBAR0_PRS4)		FC00_8400h (XBAR1_PRS4)	
FC00_4500h (XBAR0_PRS5)		FC00_8500h (XBAR1_PRS5)	
FC00_4600h (XBAR0_PRS6)		FC00_8600h (XBAR1_PRS6)	
FC00_4700h (XBAR0_PRS7)		FC00_8700h (XBAR1_PRS7)	

Address:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	R	0	M7		0	M6		0	M5		0	M4				
	W															
Reset	See Chip Configuration section for this device's reset values.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	M3		0	M2		0	M1		0	M0				
	W															
Reset	See Chip Configuration section for this device's reset values.															

Table 245. XBARx_PRSn field descriptions

Field	Description
0 Reserved	This read-only bit is reserved and always has the value zero.
1–3 M7	Master 7 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
4 Reserved	This read-only bit is reserved and always has the value zero.
5–7 M6	Master 6 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
8 Reserved	This read-only bit is reserved and always has the value zero.
9–11 M5	Master 5 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
12 Reserved	This read-only bit is reserved and always has the value zero.
13–15 M4	Master 4 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
16 Reserved	This read-only bit is reserved and always has the value zero.

Table 245. XBARx_PRSn field descriptions (continued)

Field	Description
17–19 M3	Master 3 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
20 Reserved	This read-only bit is reserved and always has the value zero.
21–23 M2	Master 2 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
24 Reserved	This read-only bit is reserved and always has the value zero.
25–27 M1	Master 1 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
28 Reserved	This read-only bit is reserved and always has the value zero.
29–31 M0	Master 0 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.

Figure 152. XBAR Control Register (XBARx_CRSn)

[illegible]

Table 246. XBARx_CRSn field descriptions

Field	Description
0 RO	Read only Forces both of the slave port's registers to be read-only. After set, only a hardware reset clears it. 0 The slave port's registers are writable. 1 The slave port's registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HRP	Halt Request Priority Determines if a request to halt the crossbar is treated as the highest priority request, or the lowest priority request. 0 The halt request has the highest priority for arbitration on this slave port. 1 The halt request has the lowest initial priority for arbitration on this slave port.
2–7 Reserved	This read-only bitfield is reserved and always has the value zero.
8–15 HPEx	High Priority Enable Determines if Master x is able to temporarily elevate its request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master x are ignored and are treated the same as regular requests. 1 High Priority requests from Master x can temporarily elevate the priority level of the master's request to the slave.
16–21 Reserved	This read-only bitfield is reserved and always has the value zero.

Table 246. XBARx_CRSn field descriptions (continued)

Field	Description
22–23 ARB	Arbitration mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority 10 Reserved 11 Reserved
24–25 Reserved	This read-only bitfield is reserved and always has the value zero.
26–27 PCTL	Parking control Determines the slave port's parking control. The low-power park feature results in overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field. 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port. 10 When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state. 11 Reserved
28 Reserved	This read-only bit is reserved and always has the value zero.
29–31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. <i>Note: Only select master ports that are actually present on the device. If not, undefined behavior may occur.</i> 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

17.2 Functional description

This section provides the functional details of the crossbar switch.

17.2.1 General operation

When a master sends an access to the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. It is possible to make single-clock (zero wait state) accesses through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Since the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply enters a wait state.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
 - An outstanding request to one slave port that has a long response time and
 - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a fixed-length burst transfer it retains control of the slave port until that transfer completes.

When a master has control of a given slave port, the crossbar returns all response information from the slave back to the requesting master.

The crossbar terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave buses). Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus even though a default master may be granted access to the slave port.

When a slave bus is being IDLEd by the crossbar, it can park the slave port on the master port indicated by the `XBARx_CRSn[PARK]`. This is done in an attempt to save the initial clock of arbitration delay that otherwise would be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low-power park mode in attempt to save power by using `XBARx_CRSn[PCTL]`.

17.2.2 Register coherency

Since the content of the registers has a real-time effect on the operation of the crossbar, it is important to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave-port-related master accesses; instead, they track only with slave accesses.

17.2.3 Arbitration

The crossbar switch supports two arbitration schemes: a simple fixed-priority comparison algorithm and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

17.2.3.1 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the `XBARx_PRSn` (priority registers). If two masters request access to a slave port, the master with the higher priority in the selected priority register gains control over the slave port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port performs an

arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port, the new requesting master is granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed-length burst transfer or a locked transfer. In this case, the new requesting master must wait until the end of the burst transfer or locked transfer before it is granted control of the slave port.

If the new requesting master's priority level is lower than the master that currently has control of the slave port, the new requesting master is forced to wait until the current master runs one of the following cycles:

- An IDLE cycle
- A non-IDLE cycle to a location other than the current slave port

17.2.3.2 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the physical master port number. This relative priority is compared to the ID (master port number) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far the ID of the requesting master is ahead of the ID of the last master.

After access is granted to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order 4, 5, and then 0.

Parking may continue to be used in a round-robin mode, but it does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

17.2.3.3 Priority assignment

Each master port needs to be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (XBARx_PRSn), the crossbar switch responds with a bus error and the registers are not updated.

17.3 Initialization/application information

No initialization is required by or for the crossbar switch. Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. Settings and priorities should be programmed to achieve maximum system performance.

18 Crossbar Integrity Checker (XBIC)

18.1 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers. For forward signals (master to slave), it is done by verifying the integrity of the attribute information using an 8-bit Error Detection Code (EDC). The EDC detects any single- or double-bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. For feedback signals (slave to master), it is done by comparing the consistence of the signals during the AHB dataphase. There are three signals from slave to master, hready, hresp0, and hresp2. If any of the master signals is different from the slave signals during dataphase, the error will be reported in the Error Status Register.

During the address phase of the system bus pipelined protocol, the XBIC generates an 8-bit EDC checkbit value for every transfer, passing these checkbits through the crossbar to the targeted slave where it is compared against the 8-bit EDC checkbits calculated from the crossbar slave attributes. The EDC is calculated for 64 bits of attribute information: 32 bits of system bus attribute control plus 32 bits associated with the processor core's decorated storage attribute. The resulting (72,64) code is a distance 4 encoding using a minimum odd weight H matrix definition. The EDC definition has been optimized for timing considerations with the system bus attribute signals exclusively using weight 3 code values and the decoration attribute using a mix of weight 3 and 5 codes.

During the data phase of the system bus pipelined protocol, the XBIC compares the three signals of hready, hresp0, hresp2 from entering the crossbar on the slave side to leaving the crossbar on the master side. Data phase is often more than one clock cycle due to wait cycles. All three signals are compared clock by clock during the whole data cycle.

18.2 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers
 - EDC (72,64) code protects against single and double bit errors
- Verification of feedback information for each data phase during crossbar transfer
 - hready, hresp0 & hresp2 are compared from entering the crossbar on the slave side to leaving the crossbar on the master side
- Error injection for testing
 - Programmable master and slave port specifiers
 - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
 - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable feedback check enable on a per-master-port basis

18.3 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown

in [Figure 153](#) and [Figure 154](#).

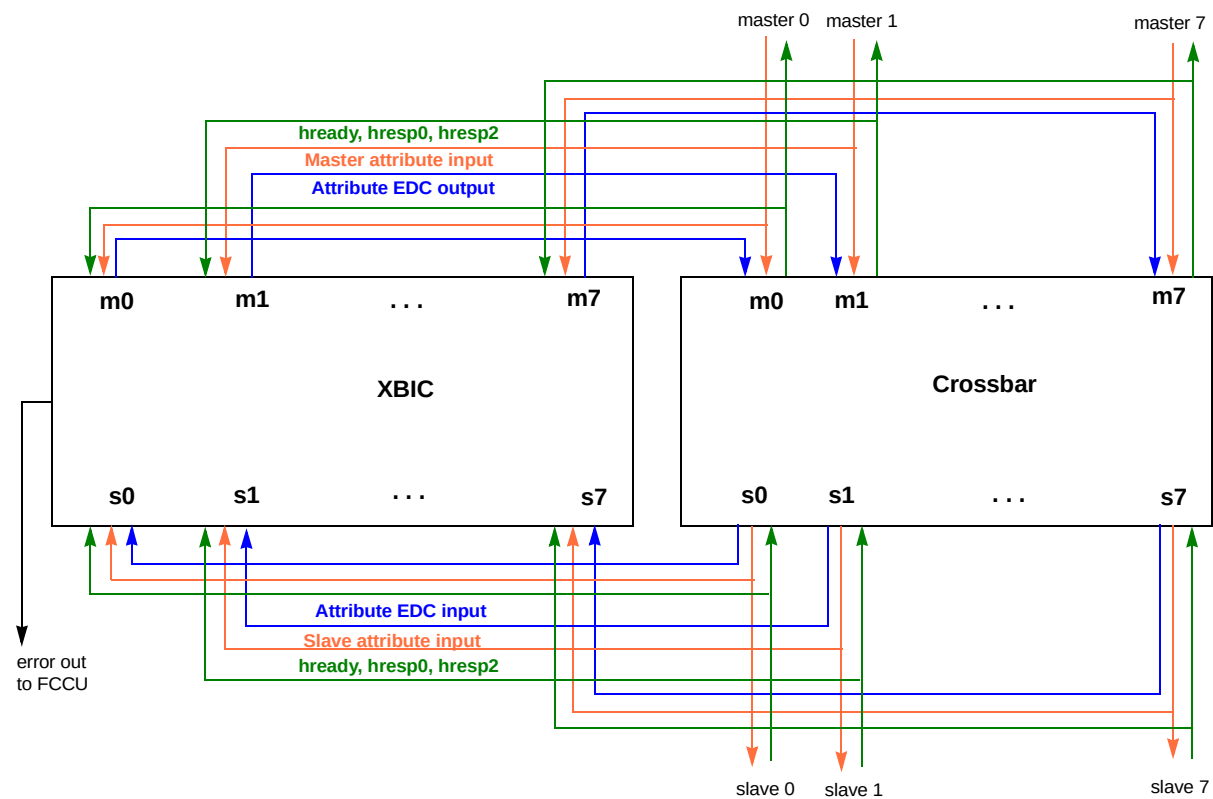


Figure 153. XBIC system block diagram

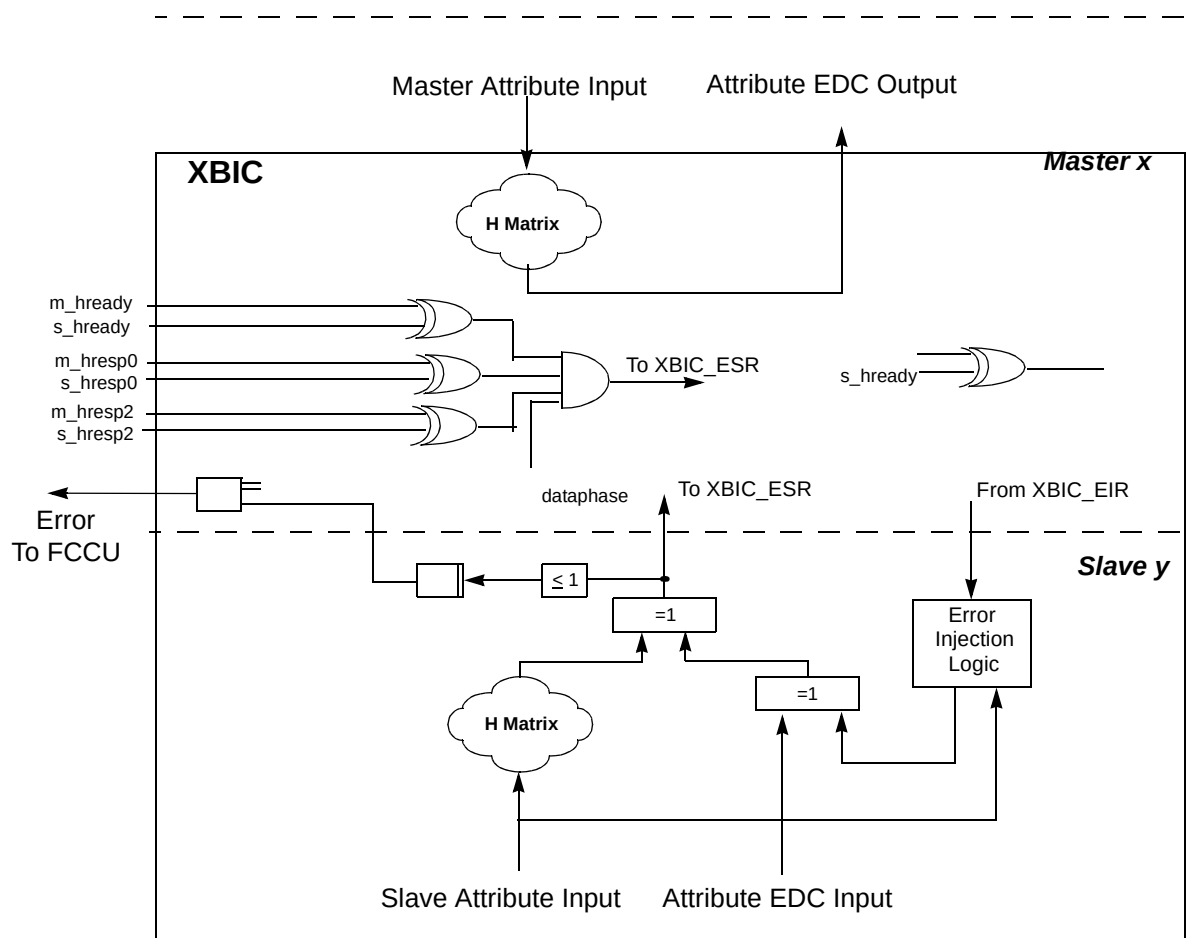


Figure 154. XBIC block diagram

18.4 External signal description

The XBIC has no external interface signals.

18.5 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references in user mode or using a different size are invalid and generate an error termination.

18.5.1 Memory map

The XBIC memory map is shown in [Table 247](#).

Table 247. XBIC memory map

Address offset	Register name	Register description	Size (bits)	Access
0x0000	XBIC_MCR	XBIC Module Control Register	32	R/W
0x0004	XBIC_EIR	XBIC Error Injection Register	32	R/W
0x0008	XBIC_ESR	XBIC Error Status Register	32	R
0x000C	XBIC_EAR	XBIC Error Address Register	32	R
0x0010 – 0x3FFF	—	Reserved	—	—

18.5.2 Register descriptions

The following sections detail the individual registers within the XBIC programming model.

18.5.2.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Figure 155. XBIC Module Control Register (XBIC_MCR)

Offset 0x0000																Access: Supervisor	
																Read/Write	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	SE0	SE1	SE2	SE3	SE4	SE5	SE6	SE7	ME0	ME1	ME2	ME3	ME4	ME5	ME6	ME7	
W																	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 248. XBIC_MCR field descriptions

Field	Description
SEn	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port n. 1 Attribute integrity checking enabled for slave port n.
MEn	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port n. 1 Attribute integrity checking enabled for master port n.

18.5.2.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Figure 156. XBIC Error Injection Register (XBIC_EIR)

Offset 0x004																Access: Supervisor	
																Read/Write	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	EIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	SLV			MST				SYN								
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 249. XBIC_EIR field descriptions

Field	Description
EIE	Error Injection Enable. 0 Error injection disabled 1 Error injection enabled
SLV	Target Slave Port. Error injection is enabled for the designated slave port. Other slave ports are unaffected.
MST	Target Master ID. Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
SYN	Syndrome. This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

18.5.2.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the *last* transfer. If an attribute integrity check error was detected, the slave port, the master port and the syndrome are captured in the SLV, MST and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are captured in the DPSE0-7 and DPME0-7 fields.

This register is cleared only on reset.

Figure 157. XBIC Error Status Register (XBIC_ESR)

Offset 0x008												Access: Supervisor Read				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	DPSE0	DPSE1	DPSE2	DPSE3	DPSE4	DPSE5	DPSE6	DPSE7	DPME0	DPME1	DPME2	DPME3	DPME4	DPME5	DPME6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DPME7	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 250. XBIC_ESR field descriptions

Field	Description
VLD	Error Status Valid. 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
DPSEn	Data phase slave port error 0) No error on slave port n 1) Data phase error on slave port n
DPMEn	Data phase master port error 0) No error on master port n 1) Data phase error on master port n
SLV	Slave Port. The slave port targeted by the last transfer with an error.
MST	Master ID. The master ID value of the last transfer with an error.
SYN	Syndrome. The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see Table 252 .

18.5.2.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the *last* transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

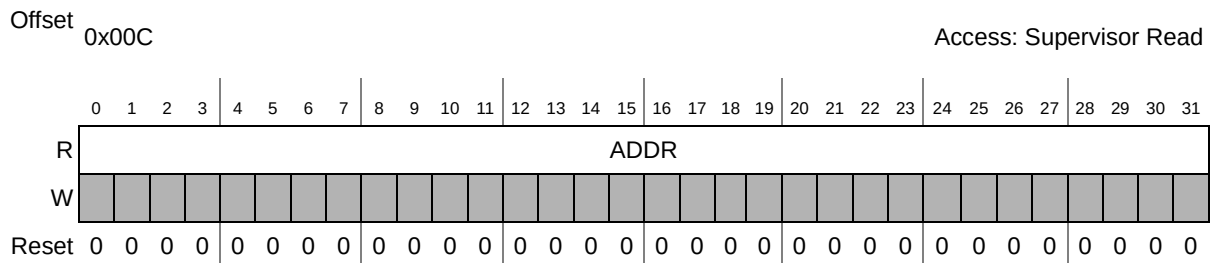


Figure 158. Error Address Register (XBIC_EAR)

Table 251. XBIC_EAR Register field descriptions

Field	Description
ADDR	The address of the last transfer with an error.

18.6 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the attribute information for crossbar transfers using an 8-bit Error Detection Code (EDC). The EDC detects any single or double bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a per-slave-port basis. The EDC parity is generated by the XBIC for each master port during the system bus address phase pipeline stage, routed through the crossbar via a sideband signal and checked against the slave attribute EDC generated for each enabled slave port during the first system bus data phase cycle. Detection of an error does *not* generate a bus error. The XBIC attribute integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

The XBIC can be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. This approach allows the check logic to be verified *without* compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared.

When an error is detected, due either to a hardware fault or error injection, information related to the error is captured and reported in the Error Status Register (XBIC_ESR) and Error Address Register (XBIC_EAR). The XBIC_ESR and XBIC_EAR registers contain information about the *last* transfer with a detected error and are only cleared on reset. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in [Table 250](#). Any other syndrome indicates a multi-bit error.

The above for XBIC to catch forwarding (master to slave) signal errors during AHB address phase. XBIC also detect backward signal (slave to master) errors during AHB data phase.

For the three slave driven signals (hready, hresp0 and hresp2), XBIC perform the comparison between the input of the slave port and output of the master port during the AHB data phase. When mis-match happening, the error will be captured. The current slave port will be registered in XBIC_ESR bit field DPSE0-7, and the current master port will be registered in XBIC_ESR bit field DSME0-7.

Table 252. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

The H matrix for the attribute EDC is shown in [Table 253](#).

Table 253. XBIC H Matrix Definition

Checkbits [7:0]	Attribute Bit																															
	hwrite	htrans[0]	hsize[2]	hsize[1]	hsize[0]	hprof[5]	hprof[4]	hprof[3]	hprof[2]	hprof[1]	hprof[0]	hburst[2]	hburst[1]	hburst[0]	hmaslock	hunalign	hbstrb[7]	hbstrb[6]	hbstrb[5]	hbstrb[4]	hbstrb[3]	hbstrb[2]	hbstrb[1]	hbstrb[0]	hmaster[3]	hmaster[2]	hmaster[1]	hmaster[0]	hslave[2]	hslave[1]	hslave[0]	hdecorated
7											*	*								*				*	*	*	*	*	*	*	*	*
6													*				*		*		*	*	*			*	*	*	*	*	*	*
5												*	*	*	*	*	*	*	*					*	*	*						*
4					*	*	*	*	*	*	*						*	*	*						*							*
3		*	*	*				*	*	*				*	*	*				*		*							*			
2	*		*	*		*	*			*			*			*					*	*	*					*				
1	*	*		*	*		*		*						*			*	*		*				*		*					
0	*	*	*		*	*		*			*	*		*				*	*		*	*			*		*					
Checkbits [7:0]	hdecor[31]	hdecor[30]	hdecor[29]	hdecor[28]	hdecor[27]	hdecor[26]	hdecor[25]	hdecor[24]	hdecor[23]	hdecor[22]	hdecor[21]	hdecor[20]	hdecor[19]	hdecor[18]	hdecor[17]	hdecor[16]	hdecor[15]	hdecor[14]	hdecor[13]	hdecor[12]	hdecor[11]	hdecor[10]	hdecor[9]	hdecor[8]	hdecor[7]	hdecor[6]	hdecor[5]	hdecor[4]	hdecor[3]	hdecor[2]	hdecor[1]	hdecor[0]
7			*	*	*	*	*	*	*		*	*	*								*	*	*				*					
6		*	*							*				*		*		*	*	*	*	*	*		*		*	*	*	*	*	*
5	*	*													*		*		*	*	*	*	*	*	*	*	*	*	*	*	*	*
4										*	*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
3		*				*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
2	*		*		*	*			*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
1			*	*		*		*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	*	*	*	*		*			*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

19 Crossbar Lock Control (CLC) module

19.1 Introduction

The crossbar switch connects bus masters and bus slaves using a hardware interconnect matrix. This structure allows all bus masters to access different bus slaves simultaneously with no interference while providing arbitration among the bus masters when they access the same slave.

The crossbar also provides a mechanism to support the lock mechanism as specified in the AMBA AHB protocol. The *hlock* signal (see [Figure 159](#)), when asserted, indicates to the crossbar that the master is performing a number of transfers to a targeted slave and the crossbar arbiter must not grant any other bus master access to the targeted slave until the locked sequence is complete. Requests by other bus masters while a slave is locked are stalled until the locked sequence is complete.

The crossbar lock control module (ACP_CLC) provides a programmable mechanism to configure a locked sequence. An instance of the ACP_CLC is associated with each crossbar switch in the device.

19.2 Block diagram

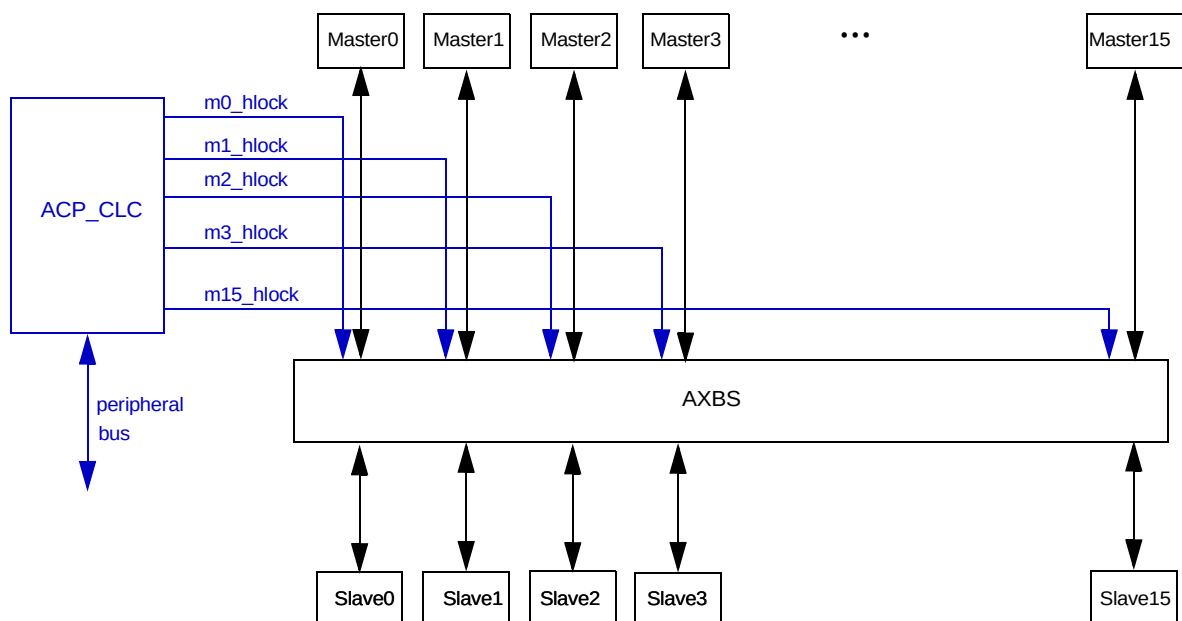


Figure 159. CLC block diagram

19.3 Memory Map and Register Definition

Each master port of the crossbar switch is provided a set of configuration registers to establish a locked transfer sequence targeting a specific slave. The registers can only be accessed in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

An error response is returned if an unimplemented location is accessed within the crossbar switch.

Note: This section shows the registers for all eight master and slave ports. If a master or slave is not used on this a device, then unexpected results occur when writing to its registers. See the Chip Configuration details for the exact master/slave assignments for your device. Additionally, all references to the crossbar switch master and slave ports are based on the physical port connections, not the logical port numbers.

Table 254. XBARx memory map

Address Offset	Register name	Width (in bits)	Access	Reset value	Section
0x00	Crossbar Lock Control Global Enable (CLC_GEN) register	32	R/W	0000_0000h	19.3.1/654
0x04	Crossbar Lock Control Master Enable (CLC_MEN) register	32	R/W	0000_0000h	19.3.2/655
0x08	Crossbar Lock Control Slave Target (CLC_STGT) register	32	R/W	0000_0000h	19.3.3/657

19.3.1 Crossbar Lock Control Global Enable (CLC_GEN) register

Figure 160. Crossbar Lock Control Global Enable (CLC_GEN) register

Address: BASE_ADDR + 0x00

Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GEN
W																
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Table 255. CLC_GEN field description

Field	Description
0–30	Reserved
31	Global Enable. This bit globally enables or disables the transfer sequence locking mechanism. 0 No locked sequence triggered by the ACP_CLC 1 Lock sequence may be triggered by the ACP_CLC <i>Note: Updates to the CLC_MEN register or CLC_STGT register will automatically clear this field.</i>

19.3.2 Crossbar Lock Control Master Enable (CLC_MEN) register

Figure 161. Crossbar Lock Control Configuration Register (CLC_MEN)

Address: BASE_ADDR + 0x04

Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M0LEN	M1LEN7	M2LEN	M3LEN	M4LEN	M5LEN	M6LEN	M7LEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 256: CLC_MEN field descriptions

Field	Description
0 M0LEN	Master 0 Lock Enable - This bit determines if the next sequence of transfers by Master 0 to the slave specified in CLC_STGT[M0SLS] are treated as a locked sequence. 0 No locked sequence triggered by Master 0 1 Lock sequence triggered by Master 0 on subsequent transfers to the slave specified in CLC_STGT[M0SLS] <i>Note: Master 0 relates to the physical crossbar port connections, not the logical master ID.</i>
1 M1LEN	Master 1 Lock Enable - This bit determines if the next sequence of transfers by Master 1 to the slave specified in CLC_STGT[M1SLS] are treated as a locked sequence. 0 No locked sequence triggered by Master 1 1 Lock sequence triggered by Master 1 on subsequent transfers to the slave specified in CLC_STGT[M1SLS] <i>Note: Master 1 relates to the physical crossbar port connections, not the logical master ID.</i>
2 M2LEN	Master 2 Lock Enable - This bit determines if the next sequence of transfers by Master 2 to the slave specified in CLC_STGT[M2SLS] are treated as a locked sequence. 0 No locked sequence triggered by Master 2 1 Lock sequence triggered by Master 2 on subsequent transfers to the slave specified in CLC_STGT[M2SLS] <i>Note: Master 2 relates to the physical crossbar port connections, not the logical master ID.</i>
3 M3LEN	Master 3 Lock Enable- This bit determines if the next sequence of transfers by Master 3 to the slave specified in CLC_STGT[M3SLS] are treated as a locked sequence. 0 No locked sequence triggered by Master 3 1 Lock sequence triggered by Master 3 on subsequent transfers to the slave specified in CLC_STGT[M3SLS] <i>Note: Master 3 relates to the physical crossbar port connections, not the logical master ID.</i>
4 M4LEN	Master 4 Lock Enable - This bit determines if the next sequence of transfers by Master 4 to the slave specified in CLC_STGT[M4SLS] are treated as a locked sequence. 0 No locked sequence triggered by Master 4 1 Lock sequence triggered by Master 4 on subsequent transfers to the slave specified in CLC_STGT[M4SLS] <i>Note: Master 4 relates to the physical crossbar port connections, not the logical master ID.</i>

Table 256: CLC_MEN field descriptions

Field	Description
5 M5LEN	Master 5 Lock Enable - This bit determines if the next sequence of transfers by Master 5 to the slave specified in CLC_STGT[M5SLS] are treated as a locked sequence. 0 No locked sequence triggered by Master 5 1 Lock sequence triggered by Master 5 on subsequent transfers to the slave specified in CLC_STGT[M5SLS] <i>Note: Master 5 relates to the physical crossbar port connections, not the logical master ID.</i>
6 M6LEN	Master 6 Lock Enable - This bit determines if the next sequence of transfers by Master 6 to the slave specified in CLC_STGT[M6SLS] are treated as a locked sequence. 0 No locked sequence triggered by Master 6 1 Lock sequence triggered by Master 6 on subsequent transfers to the slave specified in CLC_STGT[M6SLS] <i>Note: Master 6 relates to the physical crossbar port connections, not the logical master ID.</i>
7 M7LEN	Master 7 Lock Enable - This bit determines if the next sequence of transfers by Master 7 to the slave specified in CLC_STGT[M7SLS] are treated as a locked sequence. 0 No locked sequence triggered by Master 7 1 Lock sequence triggered by Master 7 on subsequent transfers to the slave specified in CLC_STGT[M7SLS] <i>Note: Master 7 relates to the physical crossbar port connections, not the logical master ID.</i>
8–31	Reserved

19.3.3 Crossbar Lock Control Slave Target (CLC_STGT) register

Figure 162. Crossbar Lock Control Slave Target (CLC_STGT) register

Address: BASE_ADDR + 0x08

Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M0STGT				M1STGT				M2STG				M3STGT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M4STGT				M5STGT				M6STGT				M7STGT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 257. CLC_STGT field description

Field	Description
0–3 M0STGT	Master 0 Slave Target. When Master 0 is configured to trigger a locked sequence, this field specifies the targeted slave to receive the locked sequence. 0000 The slave connected to crossbar slave port 0 is the target for the Master 0 locked sequence. 0001 The slave connected to crossbar slave port 1 is the target for the Master 0 locked sequence. 0010 The slave connected to crossbar slave port 2 is the target for the Master 0 locked sequence. 0011 The slave connected to crossbar slave port 3 is the target for the Master 0 locked sequence. 0100 The slave connected to crossbar slave port 4 is the target for the Master 0 locked sequence. 0101 The slave connected to crossbar slave port 5 is the target for the Master 0 locked sequence. 0110 The slave connected to crossbar slave port 6 is the target for the Master 0 locked sequence. 0111 The slave connected to crossbar slave port 7 is the target for the Master 0 locked sequence. 1---- Reserved <i>Note: Master 0 relates to the physical crossbar port connections, not the logical master ID.</i>
4–7 M1STGT	Master 1 Slave Target. When Master 1 is configured to trigger a locked sequence, this field specifies the targeted slave to receive the locked sequence. 0000 The slave connected to crossbar slave port 0 is the target for the Master 1 locked sequence. 0001 The slave connected to crossbar slave port 1 is the target for the Master 1 locked sequence. 0010 The slave connected to crossbar slave port 2 is the target for the Master 1 locked sequence. 0011 The slave connected to crossbar slave port 3 is the target for the Master 1 locked sequence. 0100 The slave connected to crossbar slave port 4 is the target for the Master 1 locked sequence. 0101 The slave connected to crossbar slave port 5 is the target for the Master 1 locked sequence. 0110 The slave connected to crossbar slave port 6 is the target for the Master 1 locked sequence. 0111 The slave connected to crossbar slave port 7 is the target for the Master 1 locked sequence. 1---- Reserved <i>Note: Master 1 relates to the physical crossbar port connections, not the logical master ID.</i>
8–11 M2STGT	Master 2 Slave Target. When Master 2 is configured to trigger a locked sequence, this field specifies the targeted slave to receive the locked sequence. 0000 The slave connected to crossbar slave port 0 is the target for the Master 2 locked sequence. 0001 The slave connected to crossbar slave port 1 is the target for the Master 2 locked sequence. 0010 The slave connected to crossbar slave port 2 is the target for the Master 2 locked sequence. 0011 The slave connected to crossbar slave port 3 is the target for the Master 2 locked sequence. 0100 The slave connected to crossbar slave port 4 is the target for the Master 2 locked sequence. 0101 The slave connected to crossbar slave port 5 is the target for the Master 2 locked sequence. 0110 The slave connected to crossbar slave port 6 is the target for the Master 2 locked sequence. 0111 The slave connected to crossbar slave port 7 is the target for the Master 2 locked sequence. 1---- Reserved <i>Note: Master 2 relates to the physical crossbar port connections, not the logical master ID.</i>

Table 257. CLC_STGT field description (continued)

Field	Description
12–15 M3STGT	Master 3 Slave Target. When Master 3 is configured to trigger a locked sequence, this field specifies the targeted slave to receive the locked sequence. 0000 The slave connected to crossbar slave port 0 is the target for the Master 3 locked sequence. 0001 The slave connected to crossbar slave port 1 is the target for the Master 3 locked sequence. 0010 The slave connected to crossbar slave port 2 is the target for the Master 3 locked sequence. 0011 The slave connected to crossbar slave port 3 is the target for the Master 3 locked sequence. 0100 The slave connected to crossbar slave port 4 is the target for the Master 3 locked sequence. 0101 The slave connected to crossbar slave port 5 is the target for the Master 3 locked sequence. 0110 The slave connected to crossbar slave port 6 is the target for the Master 3 locked sequence. 0111 The slave connected to crossbar slave port 7 is the target for the Master 3 locked sequence. 1---- Reserved <i>Note: Master 3 relates to the physical crossbar port connections, not the logical master ID.</i>
16–19 M4STGT	Master 4 Slave Target. When Master 4 is configured to trigger a locked sequence, this field specifies the targeted slave to receive the locked sequence. 0000 The slave connected to crossbar slave port 0 is the target for the Master 4 locked sequence. 0001 The slave connected to crossbar slave port 1 is the target for the Master 4 locked sequence. 0010 The slave connected to crossbar slave port 2 is the target for the Master 4 locked sequence. 0011 The slave connected to crossbar slave port 3 is the target for the Master 4 locked sequence. 0100 The slave connected to crossbar slave port 4 is the target for the Master 4 locked sequence. 0101 The slave connected to crossbar slave port 5 is the target for the Master 4 locked sequence. 0110 The slave connected to crossbar slave port 6 is the target for the Master 4 locked sequence. 0111 The slave connected to crossbar slave port 7 is the target for the Master 4 locked sequence. 1---- Reserved <i>Note: Master 4 relates to the physical crossbar port connections, not the logical master ID.</i>
20–23 M5STGT	Master 5 Slave Target. When Master 5 is configured to trigger a locked sequence, this field specifies the targeted slave to receive the locked sequence. 0000 The slave connected to crossbar slave port 0 is the target for the Master 5 locked sequence. 0001 The slave connected to crossbar slave port 1 is the target for the Master 5 locked sequence. 0010 The slave connected to crossbar slave port 2 is the target for the Master 5 locked sequence. 0011 The slave connected to crossbar slave port 3 is the target for the Master 5 locked sequence. 0100 The slave connected to crossbar slave port 4 is the target for the Master 5 locked sequence. 0101 The slave connected to crossbar slave port 5 is the target for the Master 5 locked sequence. 0110 The slave connected to crossbar slave port 6 is the target for the Master 5 locked sequence. 0111 The slave connected to crossbar slave port 7 is the target for the Master 5 locked sequence. 1---- Reserved <i>Note: Master 5 relates to the physical crossbar port connections, not the logical master ID.</i>

Table 257. CLC_STGT field description (continued)

Field	Description
24–27 M6STGT	Master 6 Slave Target. When Master 6 is configured to trigger a locked sequence, this field specifies the targeted slave to receive the locked sequence. 0000 The slave connected to crossbar slave port 0 is the target for the Master 6 locked sequence. 0001 The slave connected to crossbar slave port 1 is the target for the Master 6 locked sequence. 0010 The slave connected to crossbar slave port 2 is the target for the Master 6 locked sequence. 0011 The slave connected to crossbar slave port 3 is the target for the Master 6 locked sequence. 0100 The slave connected to crossbar slave port 4 is the target for the Master 6 locked sequence. 0101 The slave connected to crossbar slave port 5 is the target for the Master 6 locked sequence. 0110 The slave connected to crossbar slave port 6 is the target for the Master 6 locked sequence. 0111 The slave connected to crossbar slave port 7 is the target for the Master 6 locked sequence. 1---- Reserved <i>Note: Master 6 relates to the physical crossbar port connections, not the logical master ID.</i>
28–31 M7STGT	Master 7 Slave Target. When Master 7 is configured to trigger a locked sequence, this field specifies the targeted slave to receive the locked sequence. 0000 The slave connected to crossbar slave port 0 is the target for the Master 7 locked sequence. 0001 The slave connected to crossbar slave port 1 is the target for the Master 7 locked sequence. 0010 The slave connected to crossbar slave port 2 is the target for the Master 7 locked sequence. 0011 The slave connected to crossbar slave port 3 is the target for the Master 7 locked sequence. 0100 The slave connected to crossbar slave port 4 is the target for the Master 7 locked sequence. 0101 The slave connected to crossbar slave port 5 is the target for the Master 7 locked sequence. 0110 The slave connected to crossbar slave port 6 is the target for the Master 7 locked sequence. 0111 The slave connected to crossbar slave port 7 is the target for the Master 7 locked sequence. 1---- Reserved <i>Note: Master 7 relates to the physical crossbar port connections, not the logical master ID.</i>

Note: The ACP_CLC locking mechanism is intended to associate a single master to each slave target. Configuring the ACP_CLC to execute locked sequences to a single slave from multiple masters will result in non-deterministic behavior.

19.4 Functional description

The Crossbar Lock Control module monitors the AHB buses that connect the various system masters to their corresponding master ports on the crossbar. When the ACP_CLC is enabled and configured to initiate a locked sequence for a given master, the ACP_CLC snoops the request activity on the master's AHB bus. Upon detection of transfer requests within the address range that corresponds to the targeted slave's memory map region, the ACP_CLC asserts the *hlock* signal and continues to assert *hlock* on all requests to the targeted slave until the locking mechanism is disabled. The lock configuration is disabled by clearing CLC_GEN[GEN], updating CLC_MEN[MxLEN] or updating CLC_STGT[MxTGT]. Once the lock configuration is cleared, the master must perform one additional, unlocked access to the target slave to relinquish the arbitration lock and allow the crossbar to resume normal arbitration.

19.5 Initialization/application information

The locking mechanism provided by the ACP_CLC should be established before executing locked code sequence. Once the ACP_CLC has been properly configured and enabled, the locking mechanism is available immediately.

Once the locked sequence is completed, the lock configuration in the ACP_CLC must be disabled, and the master must perform one additional, unlocked access to the target slave to relinquish the arbitration lock and allow the crossbar to resume normal arbitration.

It is strongly suggested that precautions should be taken in software to avoid locking the peripheral bridge slave. Such a risk may cause the ACP_CLC programming model to become inaccessible, causing deadlock in the system if the locking mechanism cannot be disabled.

20 Peripheral Bridge

20.1 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access a majority of peripherals on the device.

The peripheral bridge occupies a 64 MB portion of the address space. (Not all peripheral slots may be used. See [Chapter 7: Device Configuration](#) or [Chapter 5: Memory Map](#) for details on slot assignment.) The bridge includes separate clock enable inputs for each of the slots, to accommodate slower peripherals.

20.1.1 Features

Key features of the peripheral bridge are:

- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Each independently configurable peripheral includes a clock enable, which allows peripherals to operate at any speed less than the system clock rate
- Programming model that provides memory protection functionality

20.1.2 General operation

The slave devices connected to the peripheral bridge are modules that contain readable/writable control and status registers. The system masters read and write these registers through the peripheral bridge. The peripheral bridge generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the peripherals. The peripheral bridge captures read data from the peripheral interface and drives it to the crossbar switch.

Each peripheral is typically allocated one 16-KB block (or slot) of the memory map. Usually, the peripheral bridge uses the size of the addressed peripheral to perform proper data byte lane routing only; no bus decomposition (dynamic bus sizing) is performed. However, there are the following exceptions:

- Aligned 64-bit reads are permitted to 32-bit slots.
- Aligned 64-bit writes may be targeted to 32-bit slots.
- Aligned or misaligned 64-bit instruction fetches may also be performed to 32-bit slots.

All other 64-bit accesses result in an error response.

20.2 Memory map/register definition

Table 258. Peripheral bridge memory map

Offset address	Register	Width (bits)	Location
0x000	Master Privilege Register A (AIPS_MPRA)	32	on page 664
0x004	Master Privilege Register B (AIPS_MPRB)	32	on page 664
0x100	Peripheral Access Control Register A (AIPS_PACRA)	32	on page 665

Table 258. Peripheral bridge memory map (continued)

Offset address	Register	Width (bits)	Location
0x104	Peripheral Access Control Register B (AIPS_PACRB)	32	on page 665
0x108	Peripheral Access Control Register C (AIPS_PACRC)	32	on page 665
0x10C	Peripheral Access Control Register D (AIPS_PACRD)	32	on page 665
0x110	Peripheral Access Control Register E (AIPS_PACRE)	32	on page 665
0x114	Peripheral Access Control Register F (AIPS_PACRF)	32	on page 665
0x118	Peripheral Access Control Register G (AIPS_PACRG)	32	on page 665
0x11C	Peripheral Access Control Register H (AIPS_PACRH)	32	on page 665
0x140	Off-platform Peripheral Access Control Register A (AIPS_OPACRA)	32	on page 666
0x144	Off-platform Peripheral Access Control Register B (AIPS_OPACRB)	32	on page 666
0x148	Off-platform Peripheral Access Control Register C (AIPS_OPACRC)	32	on page 666
0x14C	Off-platform Peripheral Access Control Register D (AIPS_OPACRD)	32	on page 666
0x150	Off-platform Peripheral Access Control Register E (AIPS_OPACRE)	32	on page 666
0x154	Off-platform Peripheral Access Control Register F (AIPS_OPACRF)	32	on page 666
0x158	Off-platform Peripheral Access Control Register G (AIPS_OPACRG)	32	on page 666
0x15C	Off-platform Peripheral Access Control Register H (AIPS_OPACRH)	32	on page 666
0x160	Off-platform Peripheral Access Control Register I (AIPS_OPACRI)	32	on page 666
0x164	Off-platform Peripheral Access Control Register J (AIPS_OPACRJ)	32	on page 666
0x168	Off-platform Peripheral Access Control Register K (AIPS_OPACRK)	32	on page 666
0x16C	Off-platform Peripheral Access Control Register L (AIPS_OPACRL)	32	on page 666
0x170	Off-platform Peripheral Access Control Register M (AIPS_OPACRM)	32	on page 666
0x174	Off-platform Peripheral Access Control Register N (AIPS_OPACRN)	32	on page 666
0x178	Off-platform Peripheral Access Control Register O (AIPS_OPACRO)	32	on page 666
0x17C	Off-platform Peripheral Access Control Register P (AIPS_OPACRP)	32	on page 666
0x180	Off-platform Peripheral Access Control Register Q (AIPS_OPACRQ)	32	on page 666
0x184	Off-platform Peripheral Access Control Register R (AIPS_OPACRR)	32	on page 666
0x188	Off-platform Peripheral Access Control Register S (AIPS_OPACRS)	32	on page 666
0x18C	Off-platform Peripheral Access Control Register T (AIPS_OPACRT)	32	on page 666
0x190	Off-platform Peripheral Access Control Register U (AIPS_OPACRU)	32	on page 666
0x194	Off-platform Peripheral Access Control Register V (AIPS_OPACRV)	32	on page 666
0x198	Off-platform Peripheral Access Control Register W (AIPS_OPACRW)	32	on page 666
0x19C	Off-platform Peripheral Access Control Register X (AIPS_OPACRX)	32	on page 666
0x1A0	Off-platform Peripheral Access Control Register Y (AIPS_OPACRY)	32	on page 666
0x1A4	Off-platform Peripheral Access Control Register Z (AIPS_OPACRZ)	32	on page 666
0x1A8	Off-platform Peripheral Access Control Register AA (AIPS_OPACRAA)	32	on page 666

Table 258. Peripheral bridge memory map (continued)

Offset address	Register	Width (bits)	Location
0x1AC	Off-platform Peripheral Access Control Register AB (AIPS_OPACRAB)	32	on page 666
0x1B0	Off-platform Peripheral Access Control Register AC (AIPS_OPACRAC)	32	on page 666
0x1B4	Off-platform Peripheral Access Control Register AD (AIPS_OPACRAD)	32	on page 666
0x1B8	Off-platform Peripheral Access Control Register AE (AIPS_OPACRAE)	32	on page 666
0x1BC	Off-platform Peripheral Access Control Register AF (AIPS_OPACRAF)	32	on page 666

20.2.1 Master Privilege Registers (AIPS_MPRA & AIPS_MPRB)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the device to the various peripherals. The register provides one field per bus master. Each master is assigned depending on its logical master number. See [Chapter 7: Device Configuration](#) for details about the master assignments to these registers.

Figure 163. Master Privilege Register A (MPRA)

Address: 0x000 (AIPS_MPRA)																Access: Supervisor read/write																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MPROT0				MPROT1				MPROT2				MPROT3				MPROT4				MPROT5				MPROT6				MPROT7			
W																																
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

Figure 164. Master Privilege Register B (MPRB)

Address: 0x004 (AIPS_MPRB)																Access: Supervisor read/write																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MPROT8				MPROT9				MPROT10				MPROT11				MPROT12				MPROT13				MPROT14				MPROT15			
W																																
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

The MPROT n field is defined as shown in [Figure 165](#).

Figure 165. MPROT n fields

	0	1	2	3
R	0	MTR	MTW	MPL
W				

Table 259. MPROT_n field descriptions

Field	Description
0	Reserved, should be cleared
1 MTR	Master trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
2 MTW	Master trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL	Master privilege level Determines how the privilege level of the master is determined. 0 Accesses from this master are forced to user mode. 1 Accesses from this master are not forced to user mode.

20.2.2 Peripheral Access Control Registers (AIPS_PACRx)

Each of the on-platform peripherals has a four-bit PACR_n field which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACRx register.

The peripheral assignments to each PACR field are defined by the memory map slot to which the peripherals are assigned. See [Chapter 5: Memory Map](#) for the assignments for your particular device. If a peripheral is absent, the corresponding PACR field is not implemented. Reads to the location return zeroes, and writes are ignored.

[Table 260](#) shows the top-level structure of the PACRs.

Table 260. PACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x0104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x0108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x010C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x0110	PACRE	PACR32	PACR33	PACR34	PACR35	PACR36	PACR37	PACR38	PACR39
0x0114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x0118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x011C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

Figure 166. Peripheral Access Control Registers (AIPS_PACRA–H)

Address: 0x100–11C (AIPS_PACRA–H)

Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PACR n				PACR($n+1$)				PACR($n+2$)				PACR($n+3$)				PACR($n+4$)				PACR($n+5$)				PACR($n+6$)				PACR($n+7$)			
W																																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Note: The reset value of PACR0 is 0101b.

Each PACR n field is defined as shown in [Figure 167](#).

Figure 167. PACR n fields

	0	1	2	3
R	0	SP	WP	TP
W				

Table 261. PACR n field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT n [MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

20.2.3 Off-Platform Peripheral Access Control Registers (AIPS_OPACRx)

Each of the off-platform peripherals has a four-bit OPACR n field which defines the access levels supported by the given module. Eight OPACRs are grouped together to form a 32-bit OPACRx register.

The peripheral assignments to each OPACR field are defined by the memory map slot to which the peripherals are assigned. See [Chapter 5: Memory Map](#) for the assignments for your particular device. If a peripheral is absent, the corresponding OPACR is not implemented. Reads to the location return zeroes and writes are ignored.

Table 260 shows the top-level structure of the OPACR registers.

Table 262. OPACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0140	OPACRA	OPACR 0	OPACR 1	OPACR 2	OPACR 3	OPACR 4	OPACR 5	OPACR 6	OPACR 7
0x0144	OPACRB	OPACR8	OPACR 9	OPACR 10	OPACR 11	OPACR 12	OPACR 13	OPACR 14	OPACR 15
0x0148	OPACRC	OPACR 16	OPACR 17	OPACR 18	OPACR 19	OPACR 20	OPACR 21	OPACR 22	OPACR 23
0x014C	OPACRD	OPACR 24	OPACR 25	OPACR 26	OPACR 27	OPACR 28	OPACR 29	OPACR 30	OPACR 31
0x0150	OPACRE	OPACR 32	OPACR 33	OPACR 34	OPACR 35	OPACR 36	OPACR 37	OPACR 38	OPACR 39
0x0154	OPACRF	OPACR 40	OPACR 41	OPACR 42	OPACR 43	OPACR 44	OPACR 45	OPACR 46	OPACR 47
0x0158	OPACRG	OPACR 48	OPACR 49	OPACR 50	OPACR 51	OPACR 52	OPACR 53	OPACR 54	OPACR 55
0x015C	OPACRH	OPACR 56	OPACR 57	OPACR 58	OPACR 59	OPACR 60	OPACR 61	OPACR 62	OPACR 63
0x0160	OPACRI	OPACR 64	OPACR 65	OPACR 66	OPACR 67	OPACR 68	OPACR 69	OPACR 70	OPACR 71
0x0164	OPACRJ	OPACR 72	OPACR 73	OPACR 74	OPACR 75	OPACR 76	OPACR 77	OPACR 78	OPACR 79
0x0168	OPACRK	OPACR 80	OPACR 81	OPACR 82	OPACR 83	OPACR 84	OPACR 85	OPACR 86	OPACR 87
0x016C	OPACRL	OPACR 88	OPACR 89	OPACR 90	OPACR 91	OPACR 92	OPACR 93	OPACR 94	OPACR 95
0x0170	OPACRM	OPACR 96	OPACR 97	OPACR 98	OPACR 99	OPACR 100	OPACR 101	OPACR 102	OPACR 103
0x0174	OPACRN	OPACR 104	OPACR 105	OPACR 106	OPACR 107	OPACR 108	OPACR 109	OPACR 110	OPACR 111
0x0178	OPACRO	OPACR 112	OPACR 113	OPACR 114	OPACR 115	OPACR 116	OPACR 117	OPACR 118	OPACR 119
0x017C	OPACRP	OPACR 120	OPACR 121	OPACR 122	OPACR 123	OPACR 124	OPACR 125	OPACR 126	OPACR 127
0x0180	OPACRQ	OPACR 128	OPACR 129	OPACR 130	OPACR 131	OPACR 132	OPACR 133	OPACR 134	OPACR 135
0x0184	OPACRR	OPACR 136	OPACR 137	OPACR 138	OPACR 139	OPACR 140	OPACR 141	OPACR 142	OPACR 143
0x0188	OPACRS	OPACR 144	OPACR 145	OPACR 146	OPACR 147	OPACR 148	OPACR 149	OPACR 150	OPACR 151

Table 262. OPACR memory map (continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x018C	OPACRT	OPACR 152	OPACR 153	OPACR 154	OPACR 155	OPACR 156	OPACR 157	OPACR 158	OPACR 159
0x0190	OPACRU	OPACR 160	OPACR 161	OPACR 162	OPACR 163	OPACR 164	OPACR 165	OPACR 166	OPACR 167
0x0194	OPACRV	OPACR 168	OPACR 169	OPACR 170	OPACR 171	OPACR 172	OPACR 173	OPACR 174	OPACR 175
0x0198	OPACRW	OPACR 176	OPACR 177	OPACR 178	OPACR 179	OPACR 180	OPACR 181	OPACR 182	OPACR 183
0x019C	OPACRX	OPACR 184	OPACR 185	OPACR 186	OPACR 187	OPACR 188	OPACR 189	OPACR 190	OPACR 191
0x01A0	OPACRY	OPACR 192	OPACR 193	OPACR 194	OPACR 195	OPACR 196	OPACR 197	OPACR 198	OPACR 199
0x01A4	OPACRZ	OPACR 200	OPACR 201	OPACR 202	OPACR 203	OPACR 204	OPACR 205	OPACR 206	OPACR 207
0x01A8	OPACRAA	OPACR 208	OPACR 209	OPACR 210	OPACR 211	OPACR 212	OPACR 213	OPACR 214	OPACR 215
0x01AC	OPACRAB	OPACR 216	OPACR 217	OPACR 218	OPACR 219	OPACR 220	OPACR 221	OPACR 222	OPACR 223
0x01B0	OPACRAC	OPACR 224	OPACR 225	OPACR 226	OPACR 227	OPACR 228	OPACR 229	OPACR 230	OPACR 231
0x01B4	OPACRAD	OPACR 232	OPACR 233	OPACR 234	OPACR 235	OPACR 236	OPACR 237	OPACR 238	OPACR 239
0x01B8	OPACRAE	OPACR 240	OPACR 241	OPACR 242	OPACR 243	OPACR 244	OPACR 245	OPACR 246	OPACR 247
0x01BC	OPACRAF	OPACR 248	OPACR 249	OPACR 250	OPACR 251	OPACR 252	OPACR 253	OPACR 254	OPACR 255

Figure 168. Off-Platform Peripheral Access Control Registers (AIPS_OPACRA–AF)

Address: 0x140–1BC (AIPS_OPACRA–AF)

Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OPACR n				OPACR ($n+1$)				OPACR ($n+2$)				OPACR ($n+3$)				OPACR ($n+4$)				OPACR ($n+5$)				OPACR ($n+6$)				OPACR ($n+7$)			
W																																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

The OPACR_n field is defined as in [Figure 169](#).

Figure 169. OPACRn fields

	0	1	2	3
R	0	SP	WP	TP
W				

Table 263. OPACRn field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROTn[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

20.3 Functional description

The peripheral bridge serves as an interface between the crossbar switch and the slave peripheral bus. It functions as a protocol translator. Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

Accesses which fall within the address space of the peripheral bridge are decoded to provide individual module selects for peripheral devices on the slave bus interface.

20.3.1 Access support

Aligned and misaligned 32-bit and 16-bit accesses, as well as byte accesses, are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. The peripheral bridge performs two slave peripheral bus transfers for allowed 64-bit transactions, to 32-bit sized peripheral slots only. All other accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted which is larger (in size) than the targeted port, an error response is generated without a peripheral access.

21 System Memory Protection Unit (SMPU)

21.1 Overview

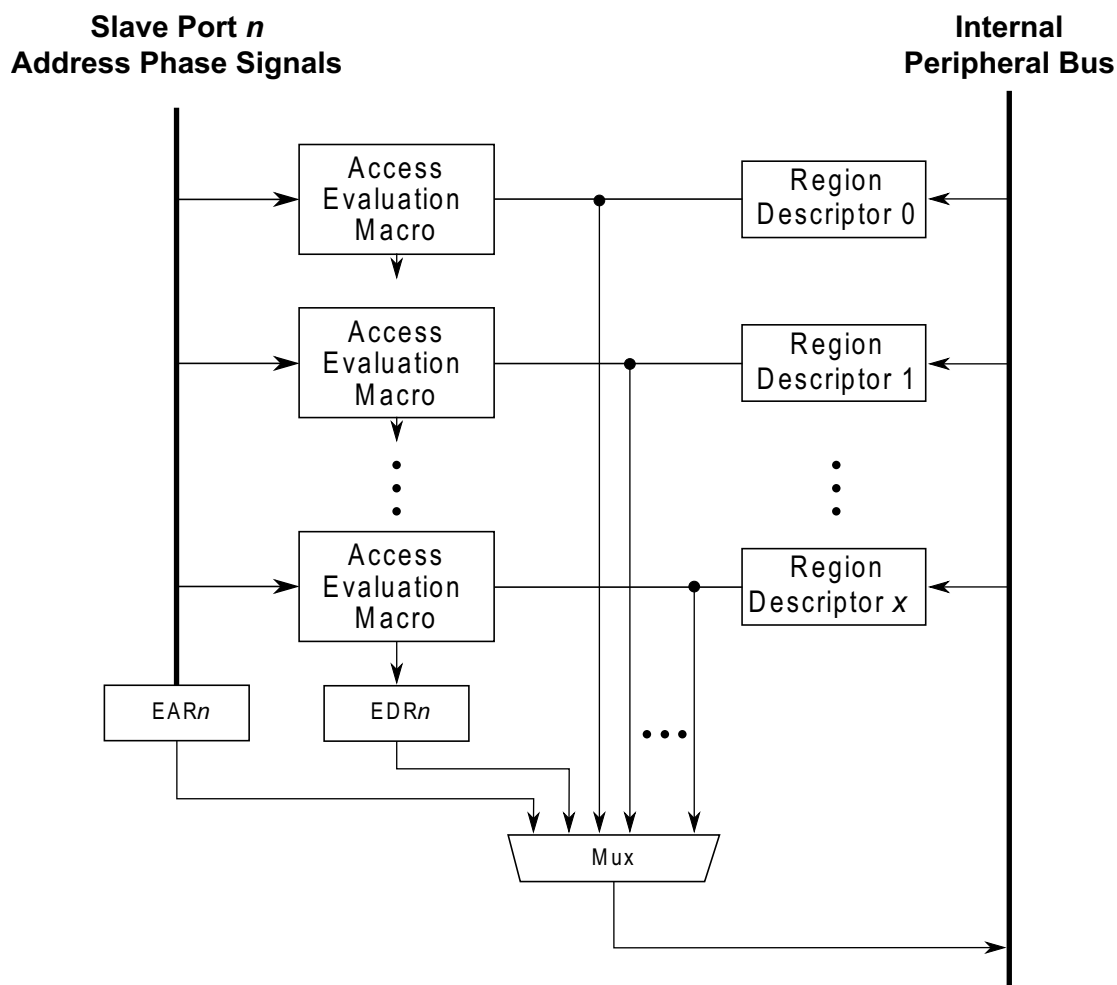
The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors and evaluates system bus transactions using pre-programmed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

21.2 Block diagram

A simplified block diagram of the SMPU module is shown in the following figure. The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers are in the middle, and the peripheral bus interface is on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and detect protection violations.

For details of the access evaluation macro, see [Section 21.5.1, Access evaluation macro](#).

Figure 170. SMPU block diagram



21.3 Features

The SMPU feature set includes:

- Supports up to 24 program-visible 128-bit region descriptors, accessible as four 32-bit words each. (Specific module instances may support fewer than 24.)
 - Each region descriptor defines an arbitrarily sized space, aligned anywhere in memory
 - Region sizes can vary from a minimum of 1 byte to a maximum of (4 GB minus 1 byte)
 - Read/write access control permissions defined in region descriptor
 - Cache-inhibit attribute indicator per region descriptor
 - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
 - Priority given to granting permission over denying access for overlapping region descriptors

- Supports as many as eight crossbar slave ports
- Detects access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Aborted core accesses generate instruction storage (ISI) or data storage (DSI) interrupts (see the documentation related to the core for details.)
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information
- Global SMPU enable/disable control bit

21.4 Memory map and register definition

21.4.1 Memory map

The programming model is partitioned into three groups: control registers, error capture registers, and the data structure containing the region descriptors.

The programming model can only be referenced using 32-bit accesses. Attempted references using different access sizes, to undefined (reserved) addresses, or with a non-supported access type (a write to a read-only register, or a read of a write-only register) generate an error termination.

The programming model can be accessed only in supervisor mode.

Note: See the device configuration details for any chip-specific register information for this module. For example, a specific instance of a module may support only up to eight region descriptors.

Table 264. SMPUx memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section
FC01_0000	SMPU0_CESR0	32	R/W	0000_8002h	on page 677
FC01_0004	SMPU0_CESR1	32	R/W	0000_8003h	on page 678
FC01_0100	SMPU0_EAR0	32	R	0000_0000h	on page 678
FC01_0104	SMPU0_EDR0	32	R	0000_0080h	on page 679
FC01_0108	SMPU0_EAR1	32	R	0000_0000h	on page 678
FC01_010C	SMPU0_EDR1	32	R	0000_0080h	on page 679
FC01_0110	SMPU0_EAR2	32	R	0000_0000h	on page 678
FC01_0114	SMPU0_EDR2	32	R	0000_0080h	on page 679
FC01_0118	SMPU0_EAR3	32	R	0000_0000h	on page 678
FC01_011C	SMPU0_EDR3	32	R	0000_0080h	on page 679
FC01_0120	SMPU0_EAR4	32	R	0000_0000h	on page 678
FC01_0124	SMPU0_EDR4	32	R	0000_0080h	on page 679
FC01_0128	SMPU0_EAR5	32	R	0000_0000h	on page 678

Table 264. SMPUx memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section
FC01_012C	SMPU0_EDR5	32	R	0000_0080h	on page 679
FC01_0130	SMPU0_EAR6	32	R	0000_0000h	on page 678
FC01_0134	SMPU0_EDR6	32	R	0000_0080h	on page 679
FC01_0138	SMPU0_EAR7	32	R	0000_0000h	on page 678
FC01_013C	SMPU0_EDR7	32	R	0000_0080h	on page 679
FC01_0140	SMPU0_EAR8	32	R	0000_0000h	on page 678
FC01_0144	SMPU0_EDR8	32	R	0000_0080h	on page 679
FC01_0148	SMPU0_EAR9	32	R	0000_0000h	on page 678
FC01_014C	SMPU0_EDR9	32	R	0000_0080h	on page 679
FC01_0150	SMPU0_EAR10	32	R	0000_0000h	on page 678
FC01_0154	SMPU0_EDR10	32	R	0000_0080h	on page 679
FC01_0158	SMPU0_EAR11	32	R	0000_0000h	on page 678
FC01_015C	SMPU0_EDR11	32	R	0000_0080h	on page 679
FC01_0160	SMPU0_EAR12	32	R	0000_0000h	on page 678
FC01_0164	SMPU0_EDR12	32	R	0000_0080h	on page 679
FC01_0168	SMPU0_EAR13	32	R	0000_0000h	on page 678
FC01_016C	SMPU0_EDR13	32	R	0000_0080h	on page 679
FC01_0170	SMPU0_EAR14	32	R	0000_0000h	on page 678
FC01_0174	SMPU0_EDR14	32	R	0000_0080h	on page 679
FC01_0178	SMPU0_EAR15	32	R	0000_0000h	on page 678
FC01_017C	SMPU0_EDR15	32	R	0000_0080h	on page 679
FC01_0400	SMPU0_RGD0_WORD0	32	R/W	0000_0000h	on page 681
FC01_0404	SMPU0_RGD0_WORD1	32	R/W	0000_0000h	on page 681
FC01_0408	SMPU0_RGD0_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_040C	SMPU0_RGD0_WORD3	32	R/W	0000_0000h	on page 683
FC01_0410	SMPU0_RGD1_WORD0	32	R/W	0000_0000h	on page 681
FC01_0414	SMPU0_RGD1_WORD1	32	R/W	0000_0000h	on page 681
FC01_0418	SMPU0_RGD1_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_041C	SMPU0_RGD1_WORD3	32	R/W	0000_0000h	on page 683
FC01_0420	SMPU0_RGD2_WORD0	32	R/W	0000_0000h	on page 681
FC01_0424	SMPU0_RGD2_WORD1	32	R/W	0000_0000h	on page 681
FC01_0428	SMPU0_RGD2_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_042C	SMPU0_RGD2_WORD3	32	R/W	0000_0000h	on page 683
FC01_0430	SMPU0_RGD3_WORD0	32	R/W	0000_0000h	on page 681

Table 264. SMPUx memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section
FC01_0434	SMPU0_RGD3_WORD1	32	R/W	0000_0000h	on page 681
FC01_0438	SMPU0_RGD3_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_043C	SMPU0_RGD3_WORD3	32	R/W	0000_0000h	on page 683
FC01_0440	SMPU0_RGD4_WORD0	32	R/W	0000_0000h	on page 681
FC01_0444	SMPU0_RGD4_WORD1	32	R/W	0000_0000h	on page 681
FC01_0448	SMPU0_RGD4_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_044C	SMPU0_RGD4_WORD3	32	R/W	0000_0000h	on page 683
FC01_0450	SMPU0_RGD5_WORD0	32	R/W	0000_0000h	on page 681
FC01_0454	SMPU0_RGD5_WORD1	32	R/W	0000_0000h	on page 681
FC01_0458	SMPU0_RGD5_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_045C	SMPU0_RGD5_WORD3	32	R/W	0000_0000h	on page 683
FC01_0460	SMPU0_RGD6_WORD0	32	R/W	0000_0000h	on page 681
FC01_0464	SMPU0_RGD6_WORD1	32	R/W	0000_0000h	on page 681
FC01_0468	SMPU0_RGD6_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_046C	SMPU0_RGD6_WORD3	32	R/W	0000_0000h	on page 683
FC01_0470	SMPU0_RGD7_WORD0	32	R/W	0000_0000h	on page 681
FC01_0474	SMPU0_RGD7_WORD1	32	R/W	0000_0000h	on page 681
FC01_0478	SMPU0_RGD7_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_047C	SMPU0_RGD7_WORD3	32	R/W	0000_0000h	on page 683
FC01_0480	SMPU0_RGD8_WORD0	32	R/W	0000_0000h	on page 681
FC01_0484	SMPU0_RGD8_WORD1	32	R/W	0000_0000h	on page 681
FC01_0488	SMPU0_RGD8_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_048C	SMPU0_RGD8_WORD3	32	R/W	0000_0000h	on page 683
FC01_0490	SMPU0_RGD9_WORD0	32	R/W	0000_0000h	on page 681
FC01_0494	SMPU0_RGD9_WORD1	32	R/W	0000_0000h	on page 681
FC01_0498	SMPU0_RGD9_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_049C	SMPU0_RGD9_WORD3	32	R/W	0000_0000h	on page 683
FC01_04A0	SMPU0_RGD10_WORD0	32	R/W	0000_0000h	on page 681
FC01_04A4	SMPU0_RGD10_WORD1	32	R/W	0000_0000h	on page 681
FC01_04A8	SMPU0_RGD10_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_04AC	SMPU0_RGD10_WORD3	32	R/W	0000_0000h	on page 683
FC01_04B0	SMPU0_RGD11_WORD0	32	R/W	0000_0000h	on page 681
FC01_04B4	SMPU0_RGD11_WORD1	32	R/W	0000_0000h	on page 681
FC01_04B8	SMPU0_RGD11_WORD2_FMT0	32	R/W	0000_0000h	on page 682

Table 264. SMPUx memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section
FC01_04BC	SMPU0_RGD11_WORD3	32	R/W	0000_0000h	on page 683
FC01_4000	SMPU1_CESR0	32	R/W	0000_8002h	on page 677
FC01_4004	SMPU1_CESR1	32	R/W	0000_8002h	on page 678
FC01_4100	SMPU1_EAR0	32	R	0000_0000h	on page 678
FC01_4104	SMPU1_EDR0	32	R	0000_0080h	on page 679
FC01_4108	SMPU1_EAR1	32	R	0000_0000h	on page 678
FC01_410C	SMPU1_EDR1	32	R	0000_0080h	on page 679
FC01_4110	SMPU1_EAR2	32	R	0000_0000h	on page 678
FC01_4114	SMPU1_EDR2	32	R	0000_0080h	on page 679
FC01_4118	SMPU1_EAR3	32	R	0000_0000h	on page 678
FC01_411C	SMPU1_EDR3	32	R	0000_0080h	on page 679
FC01_4120	SMPU1_EAR4	32	R	0000_0000h	on page 678
FC01_4124	SMPU1_EDR4	32	R	0000_0080h	on page 679
FC01_4128	SMPU1_EAR5	32	R	0000_0000h	on page 678
FC01_412C	SMPU1_EDR5	32	R	0000_0080h	on page 679
FC01_4130	SMPU1_EAR6	32	R	0000_0000h	on page 678
FC01_4134	SMPU1_EDR6	32	R	0000_0080h	on page 679
FC01_4138	SMPU1_EAR7	32	R	0000_0000h	on page 678
FC01_413C	SMPU1_EDR7	32	R	0000_0080h	on page 679
FC01_4140	SMPU1_EAR8	32	R	0000_0000h	on page 678
FC01_4144	SMPU1_EDR8	32	R	0000_0080h	on page 679
FC01_4148	SMPU1_EAR9	32	R	0000_0000h	on page 678
FC01_414C	SMPU1_EDR9	32	R	0000_0080h	on page 679
FC01_4150	SMPU1_EAR10	32	R	0000_0000h	on page 678
FC01_4154	SMPU1_EDR10	32	R	0000_0080h	on page 679
FC01_4158	SMPU1_EAR11	32	R	0000_0000h	on page 678
FC01_415C	SMPU1_EDR11	32	R	0000_0080h	on page 679
FC01_4160	SMPU1_EAR12	32	R	0000_0000h	on page 678
FC01_4164	SMPU1_EDR12	32	R	0000_0080h	on page 679
FC01_4168	SMPU1_EAR13	32	R	0000_0000h	on page 678
FC01_416C	SMPU1_EDR13	32	R	0000_0080h	on page 679
FC01_4170	SMPU1_EAR14	32	R	0000_0000h	on page 678
FC01_4174	SMPU1_EDR14	32	R	0000_0080h	on page 679
FC01_4178	SMPU1_EAR15	32	R	0000_0000h	on page 678

Table 264. SMPUx memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section
FC01_417C	SMPU1_EDR15	32	R	0000_0080h	on page 679
FC01_4400	SMPU1_RGD0_WORD0	32	R/W	0000_0000h	on page 681
FC01_4404	SMPU1_RGD0_WORD1	32	R/W	0000_0000h	on page 681
FC01_4408	SMPU1_RGD0_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_440C	SMPU1_RGD0_WORD3	32	R/W	0000_0000h	on page 683
FC01_4410	SMPU1_RGD1_WORD0	32	R/W	0000_0000h	on page 681
FC01_4414	SMPU1_RGD1_WORD1	32	R/W	0000_0000h	on page 681
FC01_4418	SMPU1_RGD1_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_441C	SMPU1_RGD1_WORD3	32	R/W	0000_0000h	on page 683
FC01_4420	SMPU1_RGD2_WORD0	32	R/W	0000_0000h	on page 681
FC01_4424	SMPU1_RGD2_WORD1	32	R/W	0000_0000h	on page 681
FC01_4428	SMPU1_RGD2_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_442C	SMPU1_RGD2_WORD3	32	R/W	0000_0000h	on page 683
FC01_4430	SMPU1_RGD3_WORD0	32	R/W	0000_0000h	on page 681
FC01_4434	SMPU1_RGD3_WORD1	32	R/W	0000_0000h	on page 681
FC01_4438	SMPU1_RGD3_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_443C	SMPU1_RGD3_WORD3	32	R/W	0000_0000h	on page 683
FC01_4440	SMPU1_RGD4_WORD0	32	R/W	0000_0000h	on page 681
FC01_4444	SMPU1_RGD4_WORD1	32	R/W	0000_0000h	on page 681
FC01_4448	SMPU1_RGD4_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_444C	SMPU1_RGD4_WORD3	32	R/W	0000_0000h	on page 683
FC01_4450	SMPU1_RGD5_WORD0	32	R/W	0000_0000h	on page 681
FC01_4454	SMPU1_RGD5_WORD1	32	R/W	0000_0000h	on page 681
FC01_4458	SMPU1_RGD5_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_445C	SMPU1_RGD5_WORD3	32	R/W	0000_0000h	on page 683
FC01_4460	SMPU1_RGD6_WORD0	32	R/W	0000_0000h	on page 681
FC01_4464	SMPU1_RGD6_WORD1	32	R/W	0000_0000h	on page 681
FC01_4468	SMPU1_RGD6_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_446C	SMPU1_RGD6_WORD3	32	R/W	0000_0000h	on page 683
FC01_4470	SMPU1_RGD7_WORD0	32	R/W	0000_0000h	on page 681
FC01_4474	SMPU1_RGD7_WORD1	32	R/W	0000_0000h	on page 681
FC01_4478	SMPU1_RGD7_WORD2_FMT0	32	R/W	0000_0000h	on page 682
FC01_447C	SMPU1_RGD7_WORD3	32	R/W	0000_0000h	on page 683

21.4.2 Register descriptions

21.4.2.1 Control/Error Status Register 0 (SMPUx_CESR0)

Figure 171. Control/Error Status Register 0 (SMPUx_CESR0)

Address:		SMPU0_CESR0 – FC01_0000h base + 0h offset = FC01_0000h												Access: Supervisor read/write			
		SMPU1_CESR0 – FC01_4000h base + 0h offset = FC01_4000h															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		MERR															
W		w1c															
Reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		1	0											HRL			GVLD
W																	
Reset		1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 265. SMPUx_CESR0 field descriptions

Field	Description
0–15 MERR	Master <i>n</i> error, where the bus master number matches the bit number Indicates a captured error in <i>EARN</i> and <i>EDRN</i> . A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing '1' to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master <i>n</i> . 1 An error has occurred for bus master <i>n</i> .
16 Reserved	This read-only bit is reserved and always has the value '1'.
17–27 Reserved	This read-only bitfield is reserved and always has the value '0'.
28–30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled

21.4.2.2 Control/Error Status Register 1 (SMPUx_CESR1)

Figure 172. Control/Error Status Register 1 (SMPUx_CESR1)

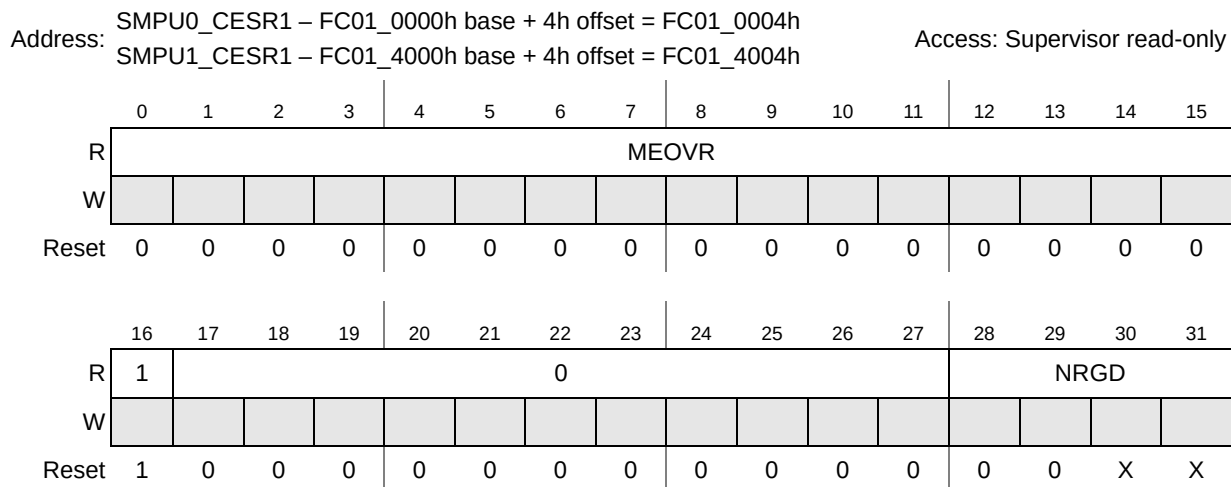


Table 266. SMPUx_CESR1 field descriptions

Field	Description
0–15 MEOVR	Master <i>n</i> error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master <i>n</i> has occurred before the previous error was processed. The details of the first error are recorded in the <i>EARN</i> and <i>EDRn</i> registers, and no information on subsequent errors is recorded until the associated <i>CESR0[MERR]</i> flag is cleared. 0 No error overrun condition has been detected for bus master <i>n</i> . 1 An error overrun condition has been detected for bus master <i>n</i> .
16 Reserved	This read-only bit is reserved and always has the value '1'.
17–27 Reserved	This read-only bitfield is reserved and always has the value '0'.
28–31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

21.4.2.3 Error Address Register, Bus Master *n* (SMPUx_EARn)

When the SMPU detects an access error on bus master *n*, the 32-bit reference address is captured in this read-only register and the corresponding bit in *CESR0[MERR]* is set. Additional information about the faulting access is captured in the corresponding *EDRn* at the same time.

Note: The corresponding $EARN$ and $EDRn$ registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the $CESR1[MEOVR]$ field until the original error is processed.

Figure 173. Error Address Register, Bus Master n (SMPUx_EARn)

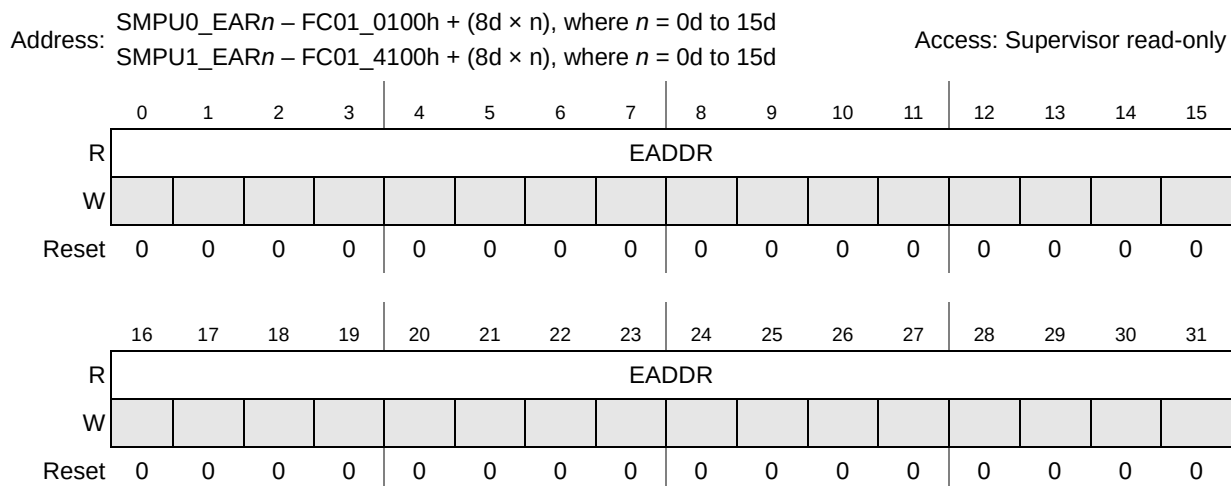


Table 267. SMPUx_EARn field descriptions

Field	Description
0–31 EADDR	Error address Indicates the reference address from bus master n that generated the access error

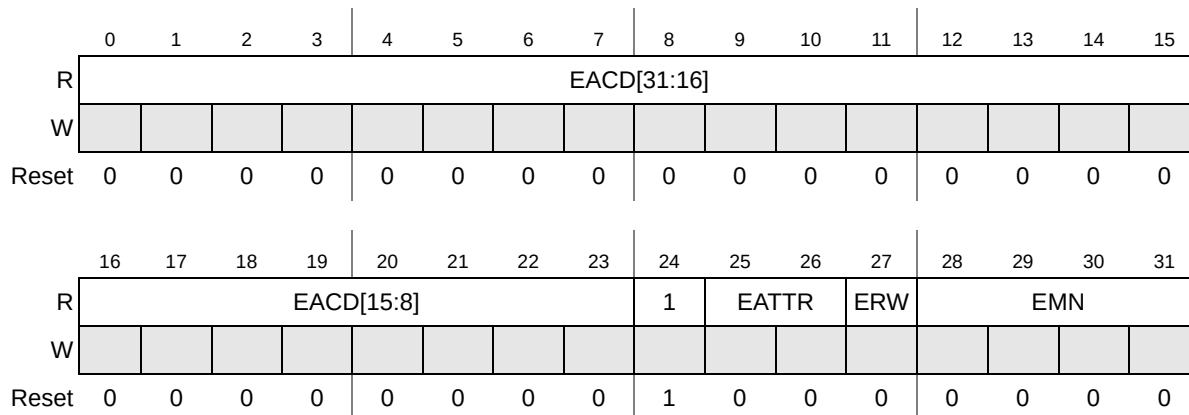
21.4.2.4 Error Detail Register, Bus Master n (SMPUx_EDRn)

When the SMPU detects an access error associated with bus master n , 32 bits of error detail are captured in this read-only register and the corresponding bit in $CESR0[MERR]$ is set. Information on the faulting address is captured in the corresponding $EARN$ register at the same time.

Note: The corresponding $EARN$ and $EDRn$ registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the $CESR1[MEOVR]$ field until the original error is processed.

Figure 174. Error Detail Register, Bus Master n (SMPUx_EDRn)

Address: SMPU0_EDRn – FC01_0104h + (8d × n), where $n = 0d$ to $15d$
 SMPU1_EDRn – FC01_4104h + (8d × n), where $n = 0d$ to $15d$ Access: Supervisor read-only

**Table 268. SMPUx_EDRn field descriptions**

Field	Description
0–23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDRn contains a captured error and EACD is all zeroes, an access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24 Reserved	This read-only bit is reserved and always has the value '1'.
25–26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
28–31 EMN	Error master number Indicates the logical bus master number of the faulting reference. This field is used to determine the bus master that generated the access error.

21.4.2.5 Region Descriptor n , Word 0 (SMPUx_RGDn_WORD0)

Figure 175. Region Descriptor n , Word 0 (SMPUx_RGDn_WORD0)

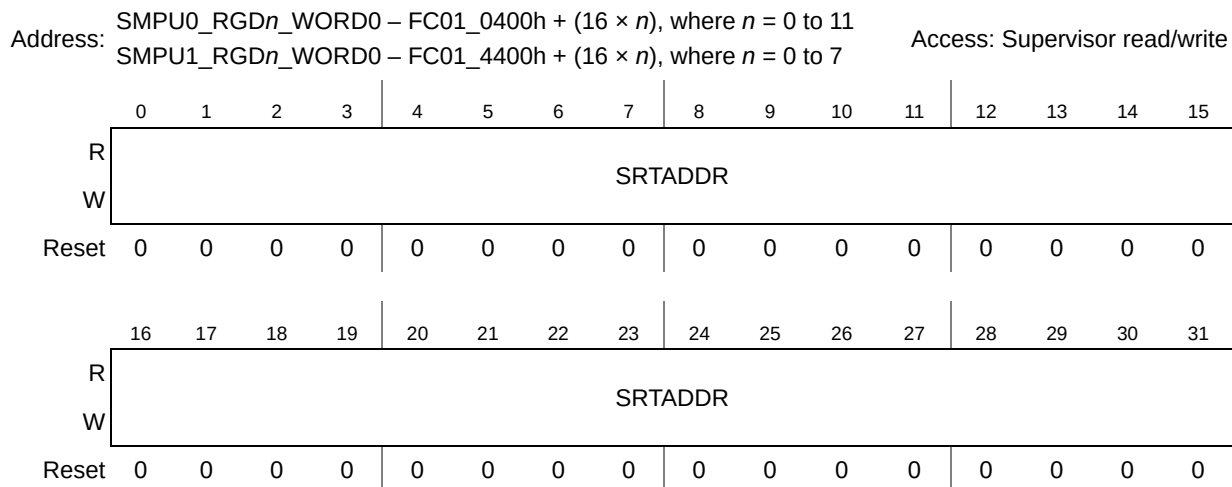


Table 269. SMPUx_RGDn_WORD0 field descriptions

Field	Description
0–31 SRTADDR	Start address Defines the byte start address of the memory region.

21.4.2.6 Region Descriptor n , Word 1 (SMPUx_RGDn_WORD1)

Figure 176. Region Descriptor n , Word 1 (SMPUx_RGDn_WORD1)

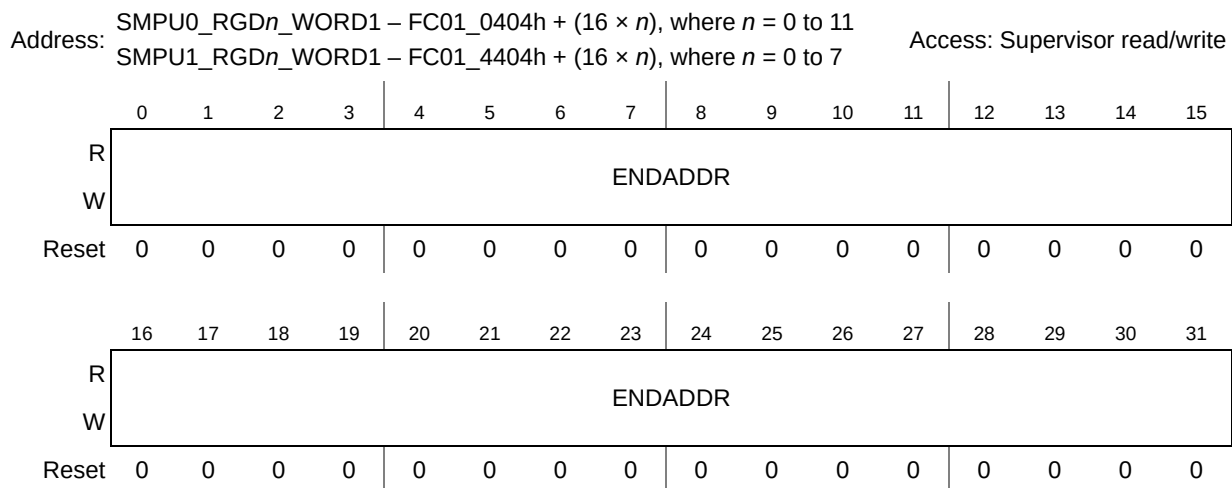


Table 270. SMPUx_RGDn_WORD1 field descriptions

Field	Description
0–31 ENDADDR	End address Defines the byte end address of the memory region. <i>Note: The SMPU does not verify that ENDADDR ≥ SRTADDR.</i>

21.4.2.7 Region Descriptor *n*, Word 2 Format 0 (SMPUx_RGDn_WORD2_FMT0)

RGD_WORD2 has two formats as determined by the RGD_WORD3[FMT] field.

Note: RGD_WORD2_FMT0 applies when RGD_WORD3[FMT] = 0.

RGD_WORD2_FMT0 defines the access control rights of the memory region on a per master basis. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags per logical bus master:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) instruction.

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Figure 177. Region Descriptor *n*, Word 2 Format 0 (SMPUx_RGDn_WORD2_FMT0)

Address: SMPU0_RGDn_WORD2_FMT0 – FC01_0408h + (16 × n), where n = 0 to 11																Access: Supervisor read/write																
SMPU1_RGDn_WORD2_FMT0 – FC01_4408h + (16 × n), where n = 0 to 7																																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15																
R	M0P				M1P				M2P				M3P				M4P				M5P				M6P				M7P			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
R	M8P				M9P				M10P				M11P				M12P				M13P				M14P				M15P			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

Table 271. SMPUx_RGDn_WORD2_FMT0 field descriptions

Field	Description
0–1 M0P	Bus master 0 permissions
2–3 M1P	Bus master 1 permissions
4–5 M2P	Bus master 2 permissions
6–7 M3P	Bus master 3 permissions
8–9 M4P	Bus master 4 permissions
10–11 M5P	Bus master 5 permissions
12–13 M6P	Bus master 6 permissions
14–15 M7P	Bus master 7 permissions
16–17 M8P	Bus master 8 permissions
18–19 M9P	Bus master 9 permissions
20–21 M10P	Bus master 10 permissions
22–23 M11P	Bus master 11 permissions
24–25 M12P	Bus master 12 permissions
26–27 M13P	Bus master 13 permissions
28–29 M14P	Bus master 14 permissions
30–31 M15P	Bus master 15 permissions

21.4.2.8 Region Descriptor *n*, Word 3 (SMPUx_RGDn_WORD3)

The final word of the SMPU region descriptor contains the valid bit, the format select (FMT), plus other configuration bits.

Figure 178. Region Descriptor n , Word 3 (SMPUX_RGD n _WORD3)

Address: SMPU0_RGD n _WORD3 – FC01_040Ch + (16 × n), where n = 0 to 11
 SMPU1_RGD n _WORD3 – FC01_440Ch + (16 × n), where n = 0 to 7

Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											FMT	RO	0	CI	VLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 272. SMPUX_RGD n _WORD3 field descriptions

Field	Description
0–26 Reserved	This read-only bitfield is reserved and always has the value '0'.
27 FMT	Region Descriptor Format This bit selects the configuration format (FMT0 or FMT1) for this region descriptor. <i>Note: A specific module instance of the SMPU may support only the FMT0 format. If so, the FMT field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.</i> 0 Use format 0 (RGD_WORD2_FMT0) 1 Use format 1 (RGD_WORD2_FMT1)
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. <i>Note: Setting RO in an RGD locks all four words of the RGD until a system reset; the valid bit of the RGD and the global valid bit have no effect.</i> 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
29 Reserved	This read-only bit is reserved and always has the value '0'.

Table 272. SMPUX_RGDn_WORD3 field descriptions (continued)

Field	Description
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. <i>Note: An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size.</i> 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals the region descriptor is valid. Any write to RGDn_WORD0–2 clears this bit. 0 Region descriptor is invalid 1 Region descriptor is valid

21.5 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

21.5.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in [Figure 179](#), the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGDn), then performs two major functions: region hit determination and detection of an access protection violation. [Figure 179](#) shows a functional block diagram.

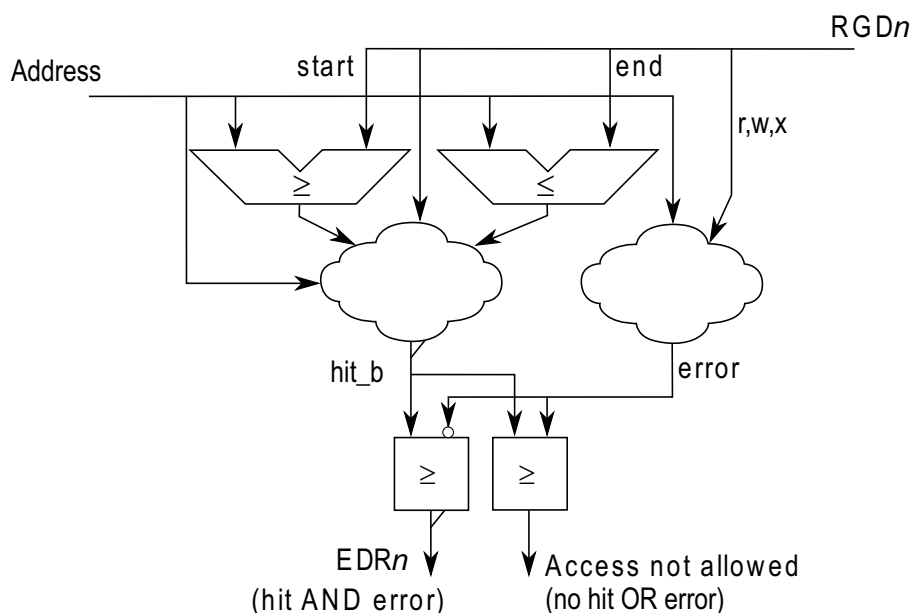


Figure 179. SMPU access evaluation macro

21.5.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. The boolean equation for this portion of the hit determination is:

```
region_hit = ((addr[0:31] >= RGDn_Word0[SRTADDR]) &
              (addr[0:31] <= RGDn_Word1[ENDADDR])) &
              RGDn_Word3[VLD]
```

where *addr* is the current reference address, *RGDn_Word0[SRTADDR]* and *RGDn_Word1[ENDADDR]* are the start and end addresses, and *RGDn_Word3[VLD]* is the valid bit.

Note: The SMPU does not verify that $ENDADDR \geq SRTADDR$.

Note: Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

21.5.1.2 Privilege violation determination

While the access evaluation macro is determining the region hit, the logic is also evaluating whether the current access is allowed by the permissions defined in the region descriptor. Using the master signal, a set of effective permissions is generated from the relevant fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions, using the specification shown in [Table 273](#).

Table 273. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

21.5.2 Putting it all together and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see

Section 21.7, Application information.

21.6 Initialization information

At system startup, load the required number of region descriptors, including setting RGDn_Word3[VLD]. Setting CESR0[GVLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0[GVLD] = 0). Next, load the appropriate region descriptors (RGDn), including the setting of the RGDn_Word3[VLD] bits. Finally, set CESR0[GVLD] to enable the entire module.

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

21.7 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities:

- Creating a new memory region—Load the appropriate region descriptor into an available RGDn, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Modifying a region descriptor—Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Removing a region descriptor—Clearing RGDn_Word3[VLD] deletes an existing region descriptor.
- Accessing the SMPU—Allocate a region descriptor to restrict SMPU access to supervisor mode from a specific master.
- Detecting an access error—The current bus cycle is terminated with an error response and EARN and EDRn capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}Rn. CESR0[MERR] signals. The error registers contain the captured fault data.
- Overlapping region descriptors—Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

The following dual-core system example contains four bus masters: the two processors (CP0, CP1) and two DMA engines (DMA1, a traditional data movement engine transferring data between RAM and peripherals, and DMA2, a second engine transferring data to/from the RAM only). Consider the following region descriptor assignments:

Table 274. Overlapping region descriptor example

Region description	RGDn		CP0	CP1	DMA1	DMA2	System memory map space
CP0 code	0		rwX	r--	—	—	Flash
CP1 code	1		r--	rwX	—	—	
CP0 data & stack	2		rw-	—	—	—	RAM
CP0 → CP1 shared data	2	3	r--	r--	—	—	
CP1 → CP0 shared data	4						
CP1 data & stack	4		—	rw-	—	—	
Shared DMA data	5		rw-	rw-	rw	rw	
SMPU	6		rw-	rw-	—	—	
Peripherals	7		rw-	rw-	rw	—	Peripheral space

In this example, there are eight descriptors used to span nine regions in the three main spaces of the system memory map (flash, RAM, and peripheral space). Each region indicates the specific permissions for each of the four bus masters, and this definition provides an assigned set of shared, private, and executable memory spaces.

Of particular interest are the two overlapping spaces: region descriptors 2 & 3 and 3 & 4.

The space defined by RGD2 with no overlap is a private data and stack area that provides read/write access to CP0 only. The overlapping space between RGD2 and RGD3 defines a shared data space for passing data from CP0 to CP1, and the access controls are defined by the logical OR of the two region descriptors. Thus, CP0 has (rw- | r--) = (rw-) permissions, while CP1 has (--- | r--) = (r--) permission in this space. Both DMA engines are excluded from this shared processor data region. The overlapping spaces between RGD3 and RGD4 defines another shared data space, this one for passing data from CP1 to CP0. For this overlapping space, CP0 has (r-- | ---) = (r--) permission, while CP1 has (rw- | r--) = (rw-) permission. The non-overlapped space of RGD4 defines a private data and stack area for CP1 only.

The space defined by RGD5 is a shared data region, accessible by all four bus masters. Finally, the slave peripheral space mapped onto the peripheral bus is partitioned into two regions: one containing the SMPU's programming model accessible only to the two processor cores and the remaining peripheral region accessible to both processors and the traditional DMA1 master.

This simple example is intended to show one possible application of the capabilities of the SMPU in a typical system.

22 Intelligent AHB Gasket (IAHBG)

22.1 Introduction

This section details the intelligent bridging gasket between the fast and slow clock domains. The intelligent operation of the gasket initiates pending reads early and optimizes bursts to recover some of the performance that is lost on a registered interface between different clock domains. Pending reads begin even if there is not an hready to the master. Non-bursted writes only begin when the slave is IDLE, pending writes wait until IDLE.

The IAHBG complies with the AHBV6lite protocol, including an extension to support masters that do address retraction during the error termination sequence or abort a burst read due to an error termination or another non-error termination like cache inhibit termination.

[Table 275](#) lists the available operation modes.

Table 275. Operation modes available

Operating mode (Master:Slave)	Comments
0:0	Flowthru (no register wall)
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

22.2 Timing diagrams

The following are a sample of timing diagrams assuming a zero wait state response. The 0:0 flowthru mode is not shown.

22.2.1 1:1

In the 1:1 timing mode the master is running at the same clock frequency as the slave. There are different read slots that the pending read transaction can begin executing early, but only the first slot is shown. The pending read can either be a single or burst transaction.

Figure 180. Read, pending read (1:1 timing mode)

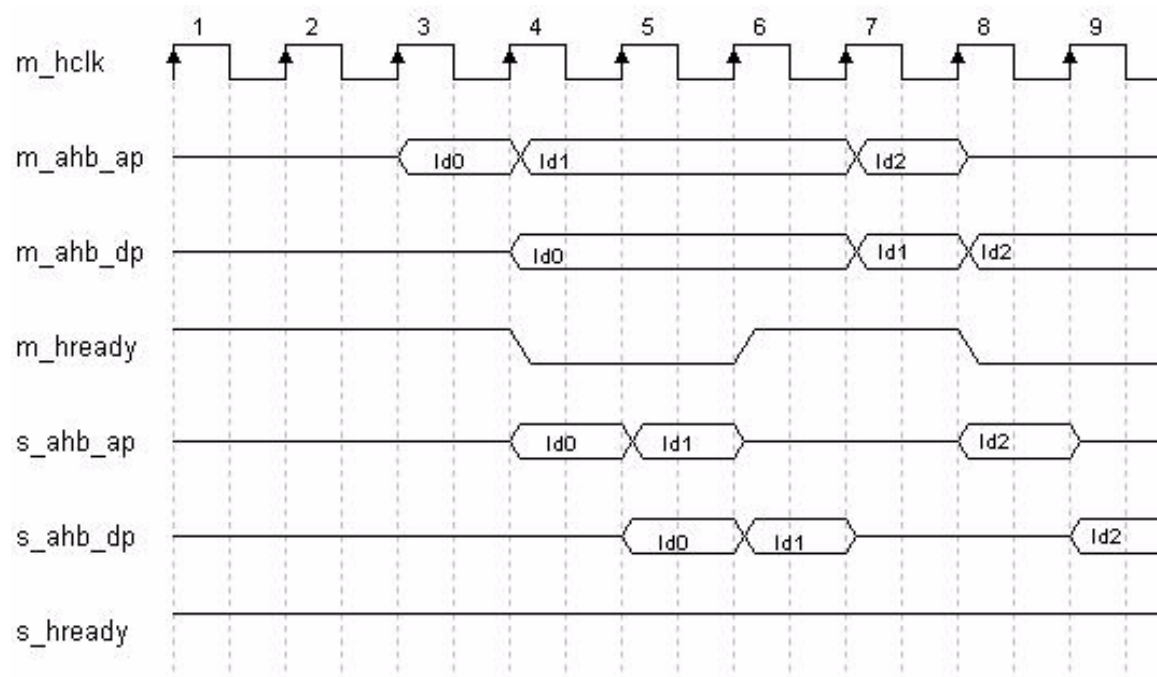


Figure 181. Write, pending read (1:1 timing mode)

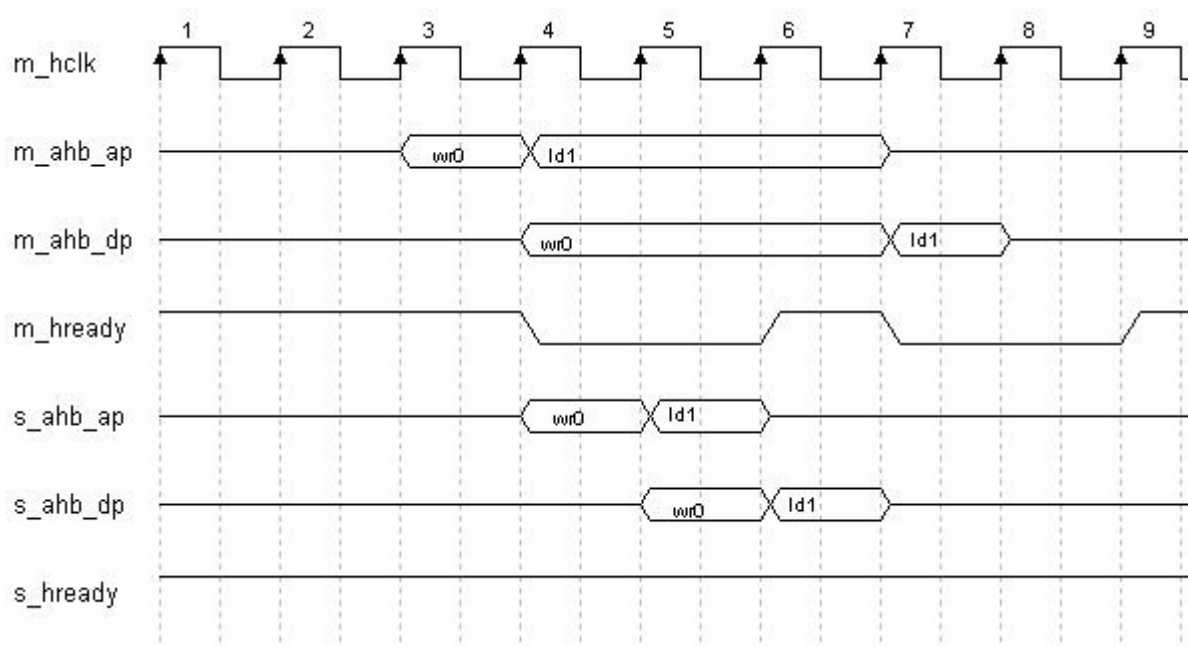


Figure 182. Burst4 read (1:1 timing mode)

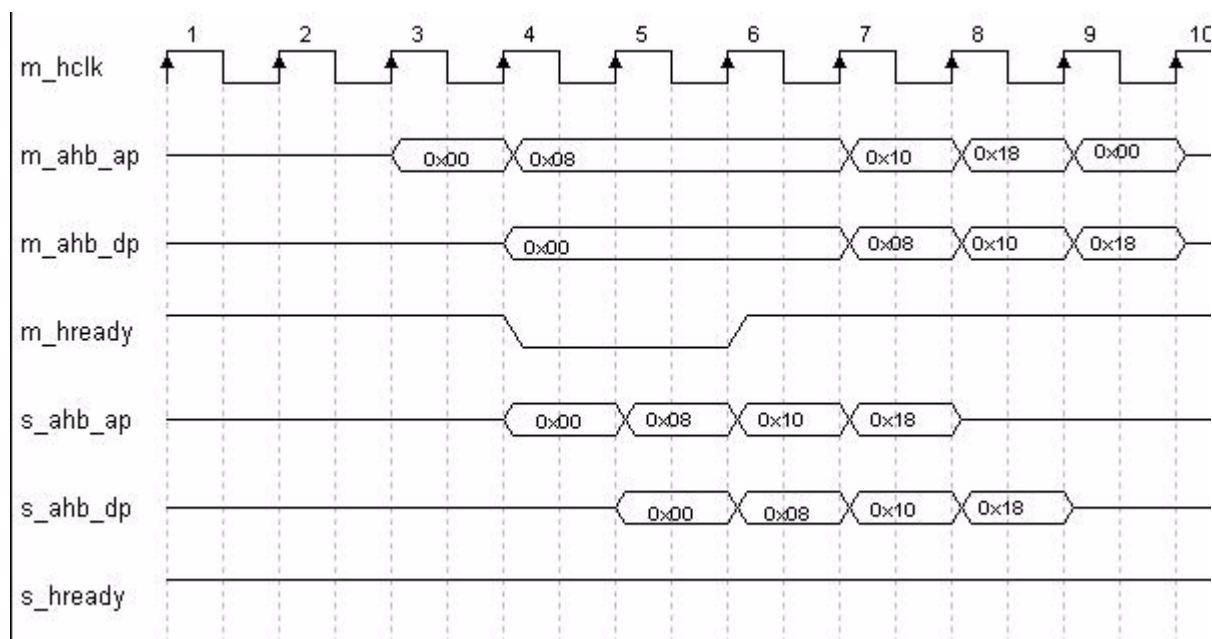
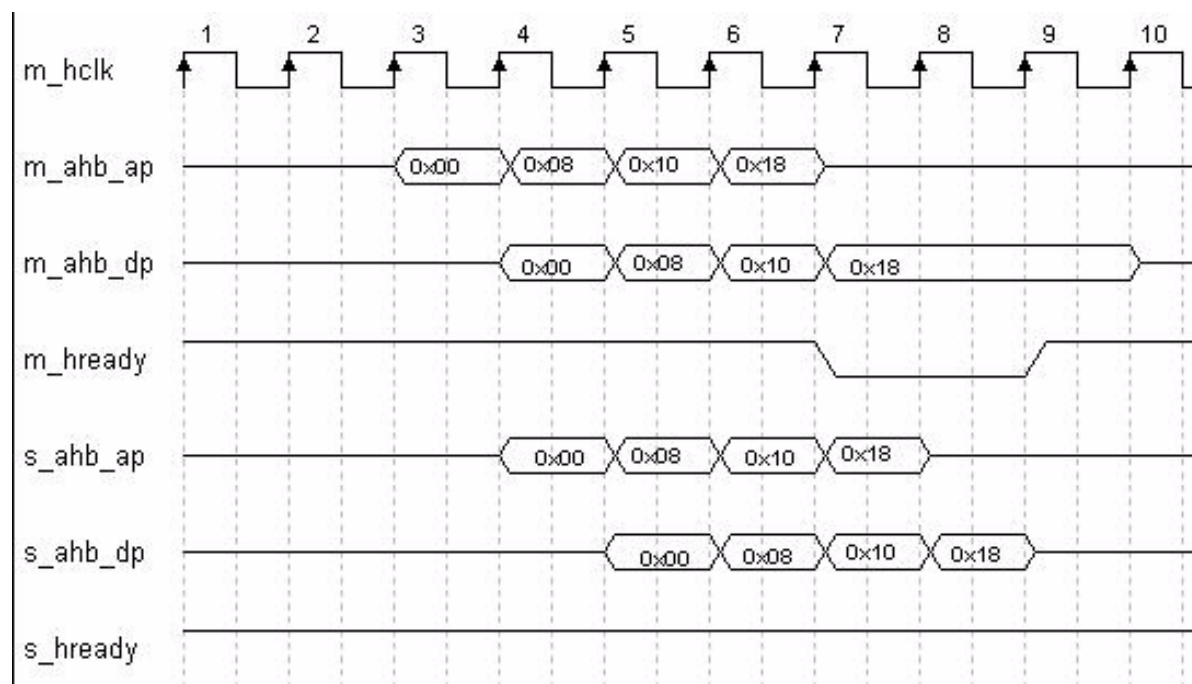


Figure 183. Burst4 write (1:1 timing mode)



22.2.2 2:1

In the 2:1 timing mode the master is running at twice the clock frequency of the slave. The 4:1 mode is not shown because it is very similar to the 2:1 mode. There are different read

slots that the pending read transaction can begin executing early, but only the first slot is shown. The pending read can either be a single or burst transaction.

Figure 184. Read, pending read (2:1 timing mode)

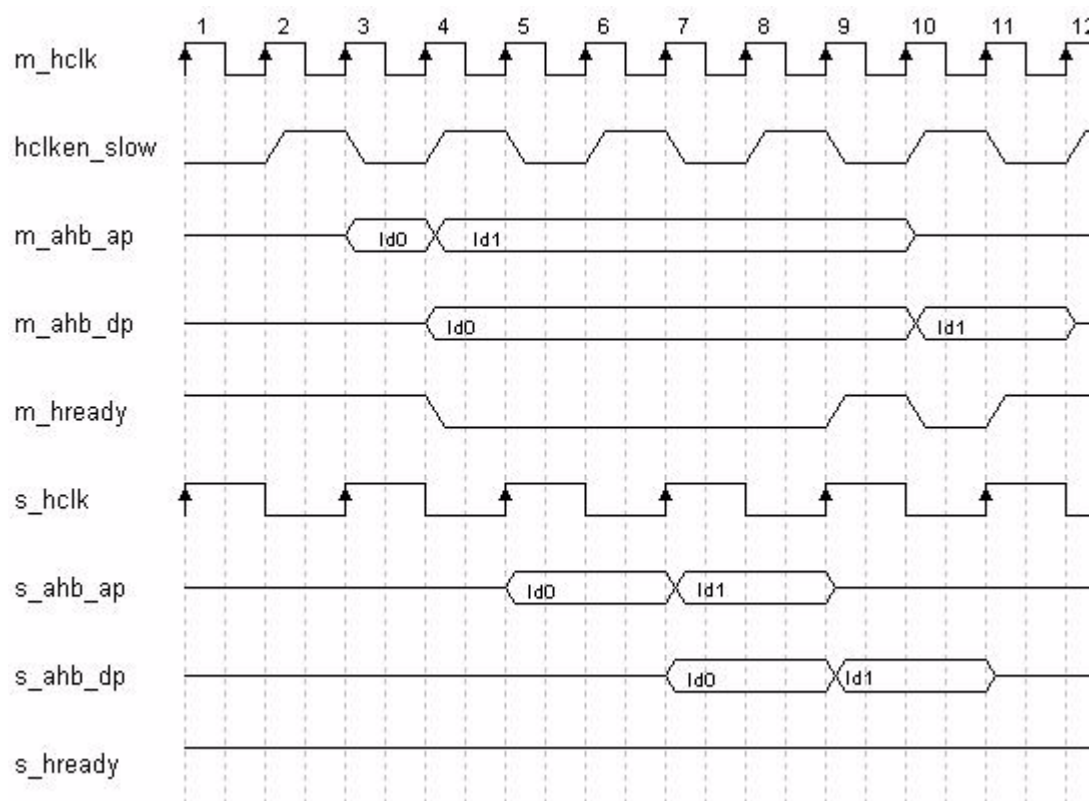


Figure 185. Write, pending read (2:1 timing mode)

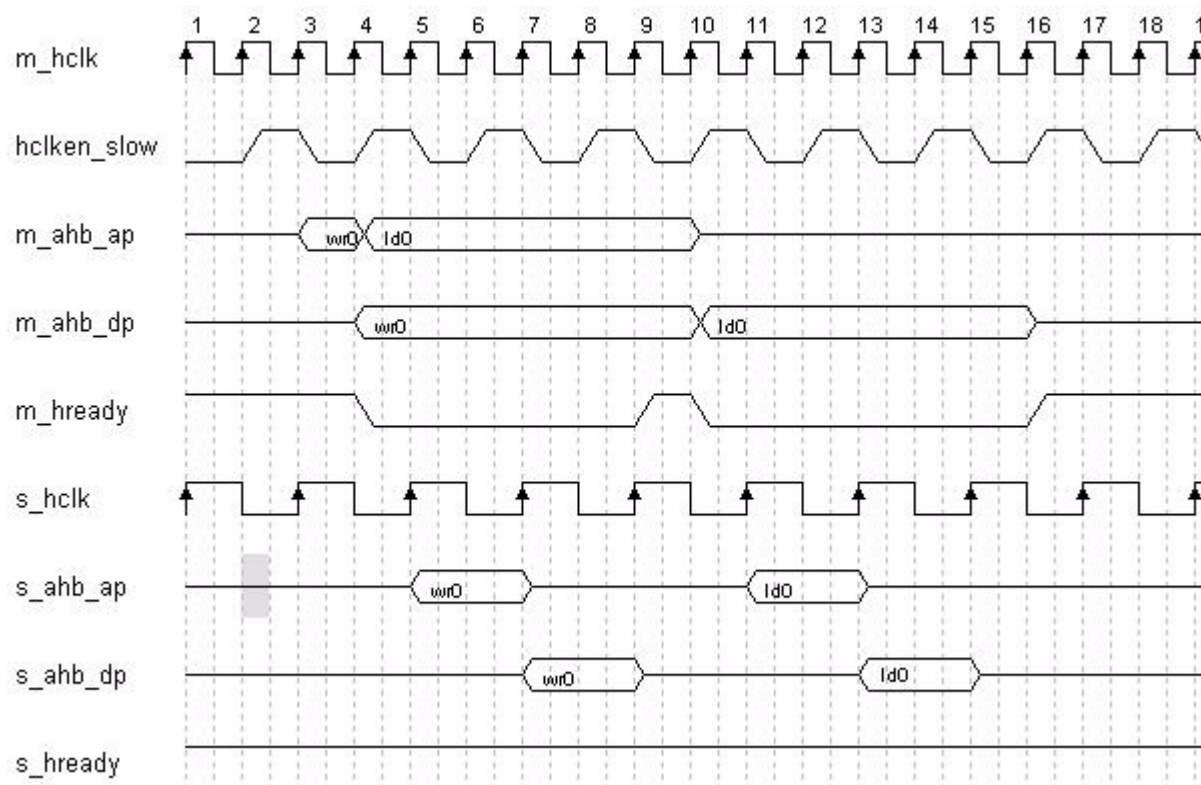


Figure 186. Burst4 read (2:1 timing mode)

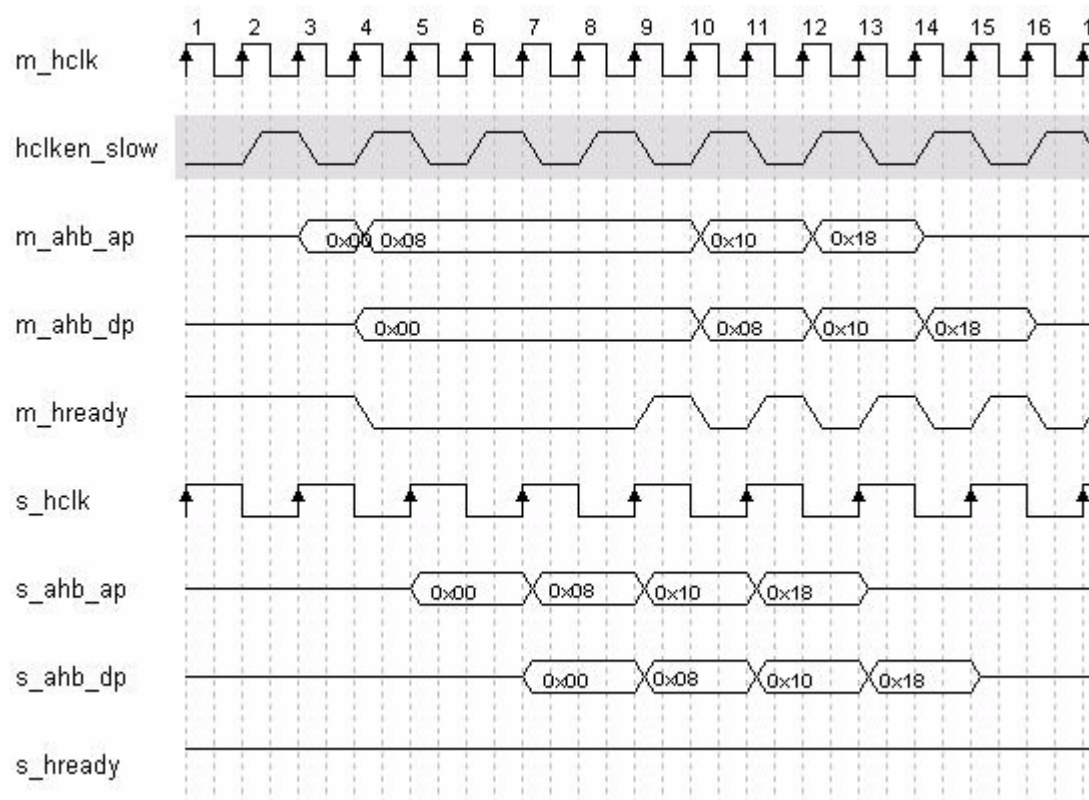
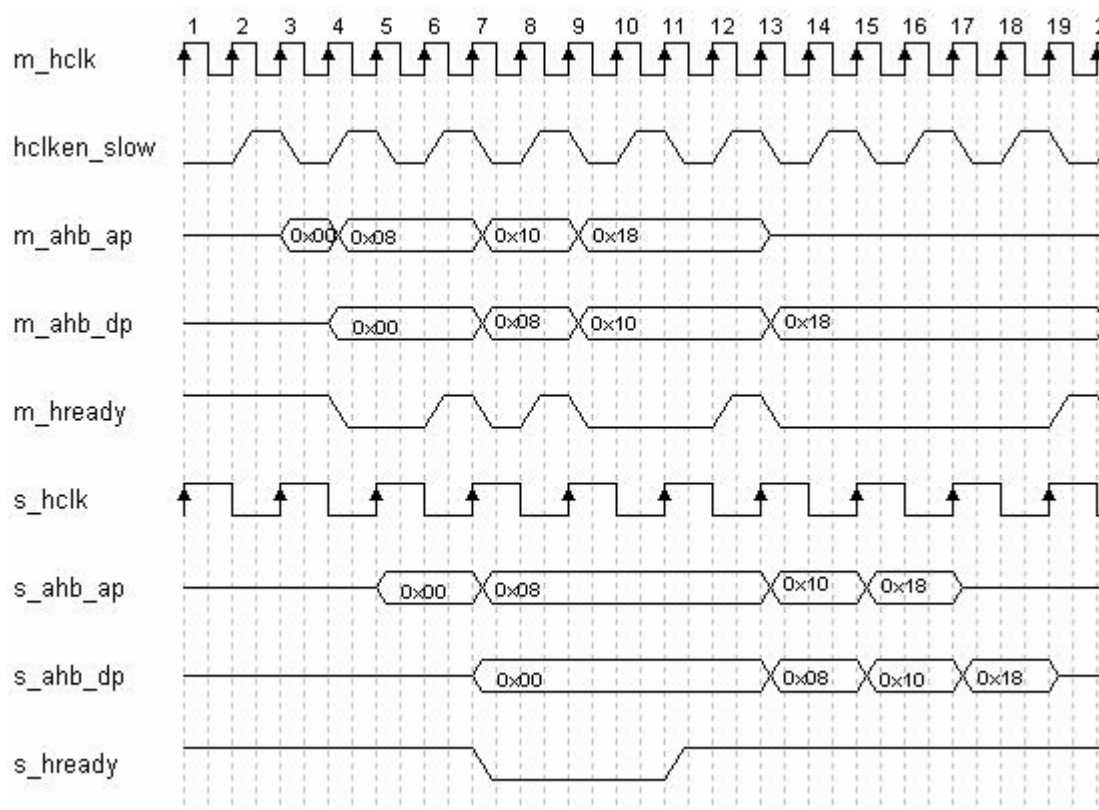


Figure 187. Burst4 write (2:1 timing mode)



22.2.3 1:2

In the 1:2 timing mode, the slave is running at twice the clock frequency of the master. The 1:4 mode is not shown because it is very similar to the 1:2 mode. There are different read slots that the pending read transaction can begin executing early, but only the first slot is shown. The pending read can either be a single or burst transaction.

Because the slave is operating at a faster clock frequency than the master, the transactions from the master must be stored into a queue of data and response attribute registers. At a certain point during the master burst transactions, the queue is either emptied into the master for a burst read or it is emptied into the slave for a burst write. Burst writes will only work across the IAHBG if an assumption is made where the hresp from the slave is accumulated into the last beat of the burst (implicit burst write mode).

Figure 188. Read, pending read (1:2 timing mode)

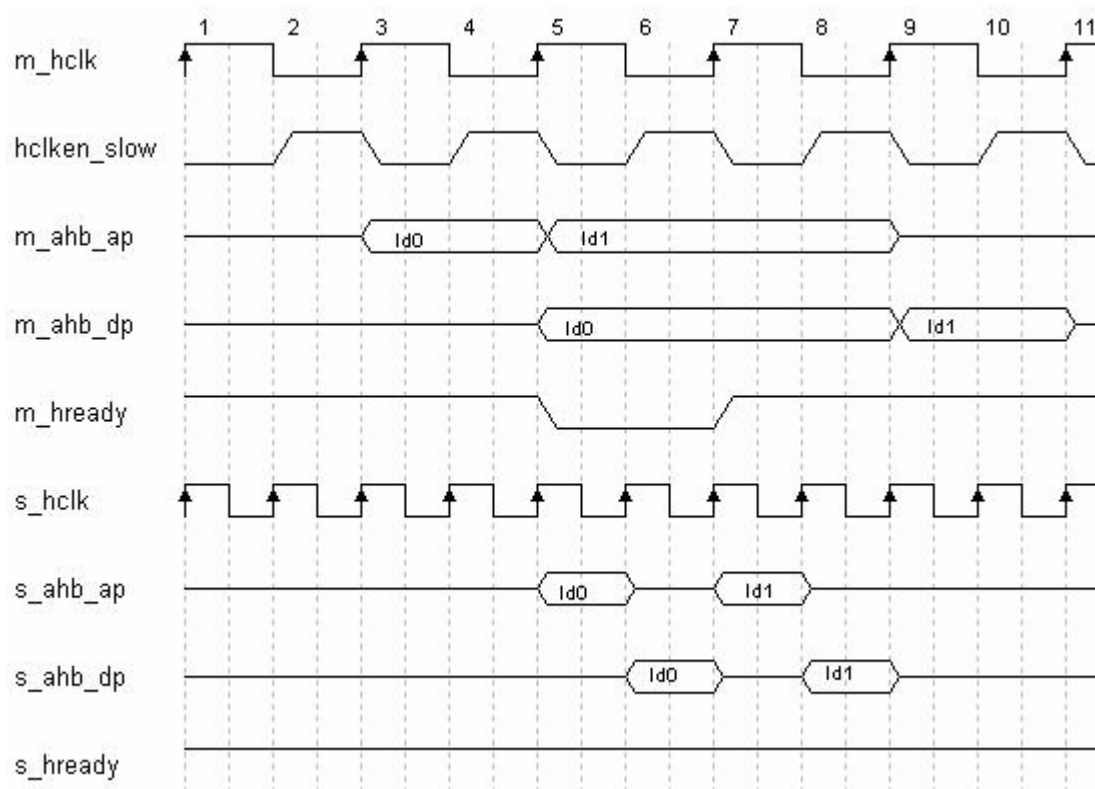


Figure 189. Write, pending read (1:2 timing mode)

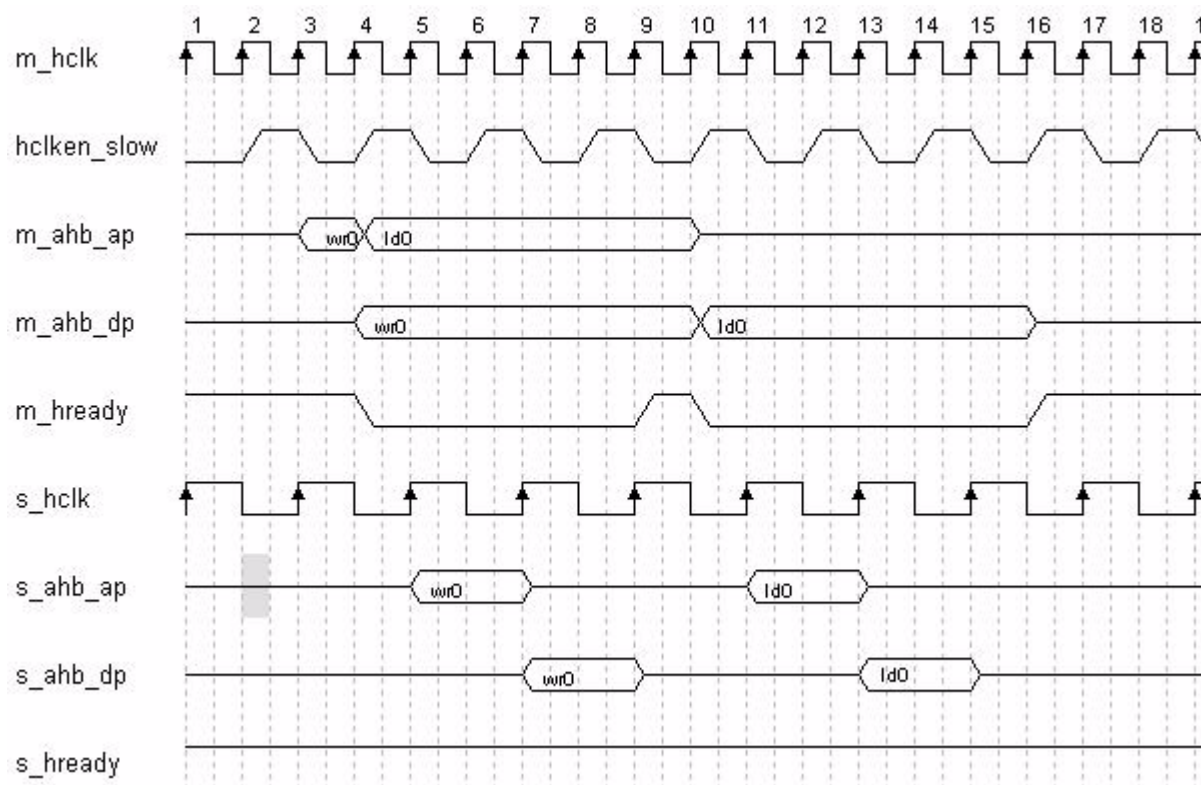


Figure 190. Burst4 read (1:2 timing mode)

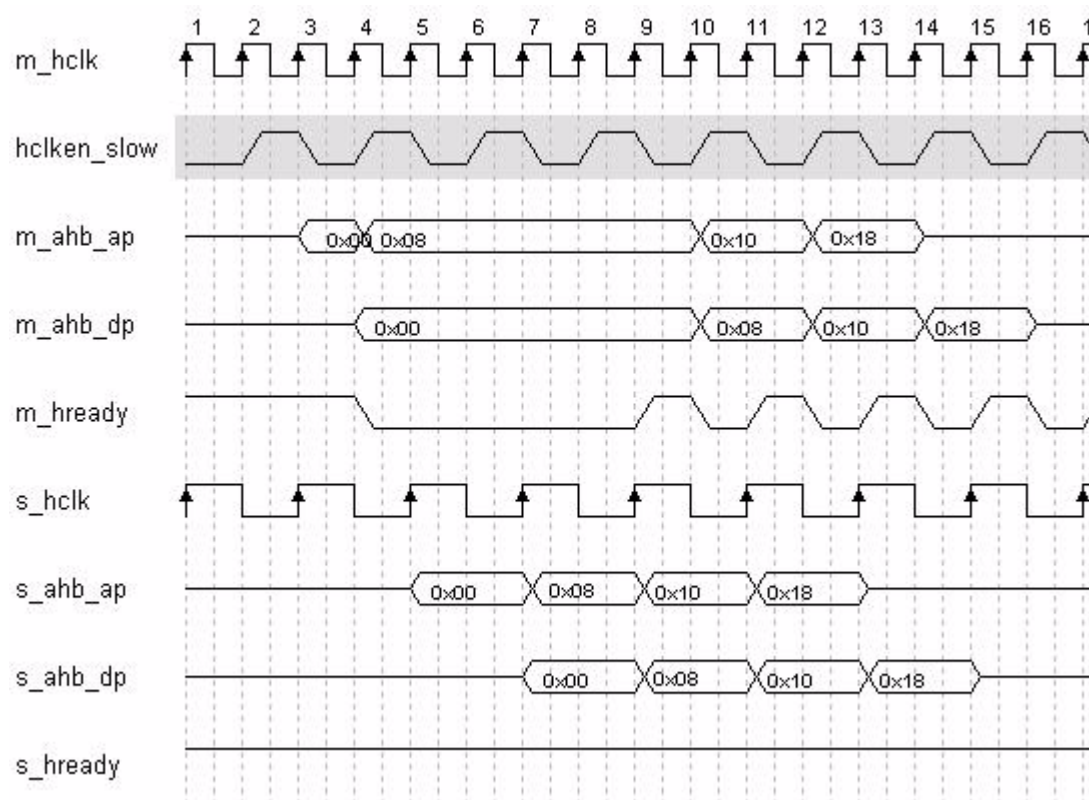
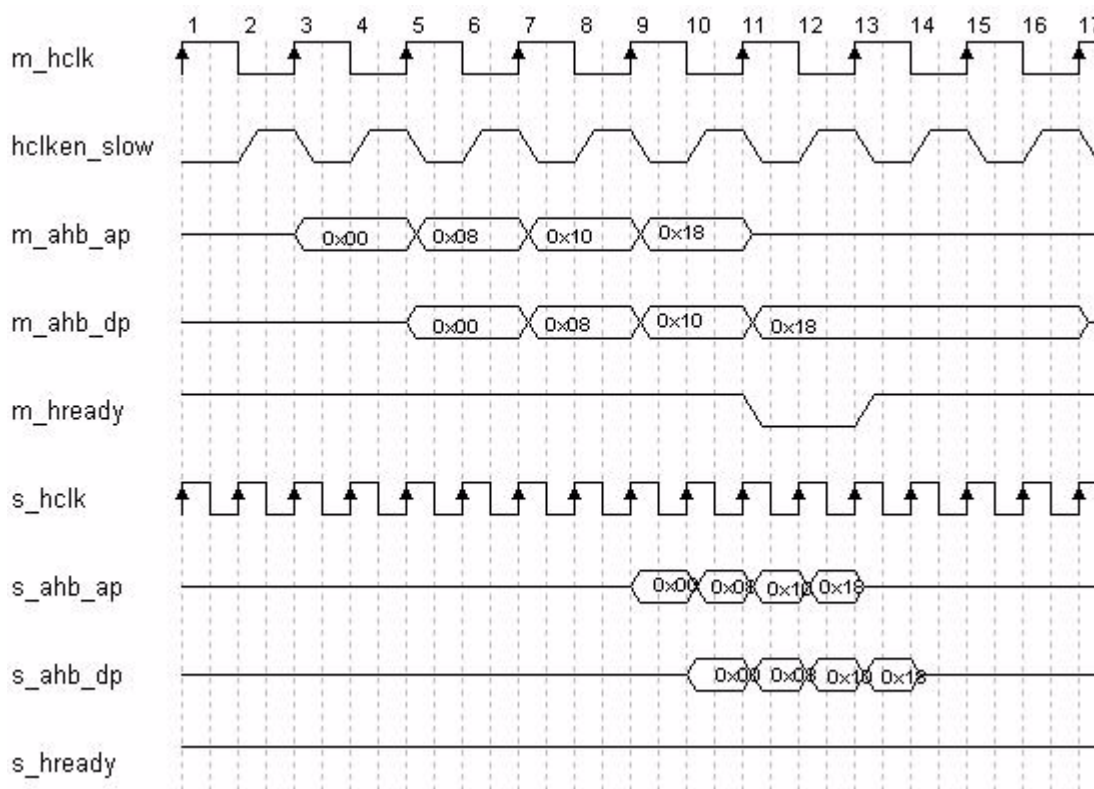


Figure 191. Burst4 write (1:2 timing mode)



22.2.4 64->32 Interface

The 64->32 interface translates a 64bit access to two 32bit accesses on the slave side. If there is an hresp on the first 32bit access to the slave, the iahb_gasket will still do the second 32-bit access. The hresp is aggregated and reported to the master in the data phase of the 64bit access. One of the 32bit writes could complete and the other write could return an hresp.

For burst encodings of WRAP4,INCR4 hsize=long, it translates to WRAP8,INCR8 hsize=word.

For burst encodings of WRAP8,INCR8 hsize=long, it translates to WRAP16,INCR16 hsize=word.

Burst encodings WRAP4,8 INCR4,8 of hsize<long are passed as is through to slave.

All other burst encodings are not optimized and are translated as NSEQ.

Unaligned bursts are not optimized and are translated as NSEQ.

22.2.5 32->64 Interface

The 32->64 interface translates two 32bit accesses to one 64bit access on the slave side.

All burst encodings are passed as is to the slave.

There is no special translation of WRAP8,INCR8 32bit to WRAP4,INCR4 64bit translation.

To support initializing ECC in the SRAM, the IOP core will send aligned (`haddr[2]==0`) double word transactions with a burst encoding of INCR with length fixed to 2. The `iahb_gasket` will translate these two 32bit writes to a single 64bit NSEQ. It will never report a `hresp` on the first 32bit write.

Burst encodings WRAP4,8 INCR4,8 of `hsize<long` are passed as is through to slave.

All other burst encodings are not optimized and are translated as NSEQ.

Unaligned bursts are translated to NSEQ.

23 Semaphores2 (SEMA42)

23.1 Introduction

The peripheral platform shell includes a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve “lock and unlock” operations via a single write access. The hardware semaphore module is an architecture-neutral solution that provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

The capabilities provided by the hardware semaphore module are currently utilized in dual-core AUTOSAR[®] implementations, and its use is expected to continue in future multi-core ports. The result is a second-generation module implementation known as Semaphores2 or SEMA42.

23.1.1 Multi-core programming 101: software gates

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism that is then used by system software to control access to shared data structures, shared hardware resources, and so on. These gating mechanisms are used by the software to serialize (and synchronize) writes to shared data and/or resources to prevent race conditions and preserve memory coherency between different processes and processors.

Consider the following description of a typical use-case. Processor X enters a section of code where shared data values are to be updated. It must first acquire a semaphore; this can be considered to be locking (or closing) a software gate. Once the gate has been locked, a properly architected software system does not allow other processes (or processors) to execute the same code segment or modify the shared data structure protected by the gate; in other words, other processes/processors are locked out. Many software implementations include a spin-wait loop within the lock function until the locking of the gate is accomplished. Once the lock has been obtained, processor X continues execution and updates the data values protected by the particular lock. Once the updates are complete, processor X unlocks (or opens) the software gate, allowing other processes/processors access to the updated data values.

There are three important rules that must be followed for a correctly implemented system solution:

1. All writes to shared data values or shared hardware resources must be protected by a “gate” variable.
2. Once a processor locks a gate, accesses to the shared data or resources by other processes/processors must be blocked. This is enforced by software conventions.
3. The processor that locks a particular gate is the only processor that can open (unlock) that gate.

Information in the hardware gate identifying the locking processor can be extremely useful for system-level debugging.

The Hennessy/Patterson text on computer architecture offers the following description of software gating.

“One of the major requirements of a shared-memory architecture multiprocessor is being able to coordinate processes that are working on a common task. Typically, a programmer will use *lock variables* to synchronize the processes.

The difficulty for the architect of a multiprocessor is to provide a mechanism to decide which processor gets the lock and to provide the operation that locks a variable. Arbitration is easy for shared-bus multiprocessors, since the bus is the only path to memory. The processor that gets the bus locks out all the other processors from memory. If the CPU and bus provide an atomic swap operation, programmers can create locks with the proper semantics. The adjective *atomic* is key, for it means that a processor can both read a location **and** set it to the locked value in the same bus operation, preventing any other processor from reading or writing memory.” [Hennessy/Patterson, *Computer Architecture: A Quantitative Approach*, ppg. 471-472]

The classic text continues with a description of the steps required to lock/unlock a variable using an *atomic swap instruction*.

“Assume that 0 means unlocked and 1 means locked. A processor first reads the lock variable to test its state. A processor keeps reading and testing until the value indicates that the lock is unlocked. The processor then races against all other processes that were similarly “spin waiting” to see who can lock the variable first. All processes use a swap instruction that reads the old value and stores a 1 into the lock variable. The single winner will see the 0, and the losers will see a 1 that was placed there by the winner. (The losers will continue to set the variable to the locked value, but this does not matter.) The winning processor executes the code after the lock and then stores a 0 into the lock when it exits, starting the race all over again. Testing the old value and then setting to a new value is why the atomic swap instruction is called *test and set* in some instruction sets.” [Hennessy/Patterson, *Computer Architecture: A Quantitative Approach*, ppg. 472-473]

Next, consider the features, programming model, and functional operation of the hardware semaphore module.

23.1.2 Features

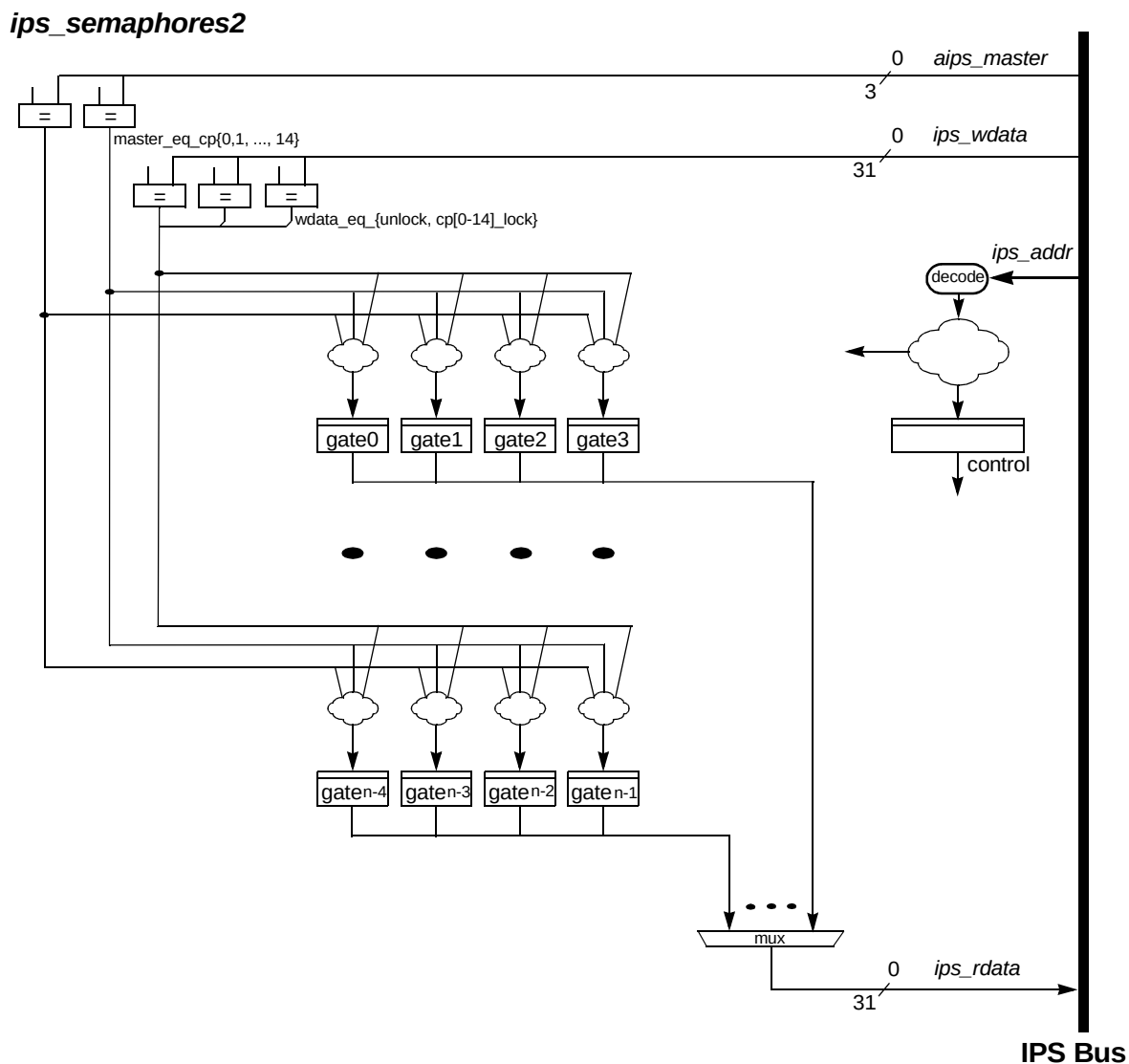
The Semaphores module implements hardware-enforced semaphores as an IPS-mapped slave peripheral device. The feature set includes:

- Module definition supporting 16, 32, or 64 hardware-enforced gates in a multi-processor configuration, where up to 15 processors can be supported; cpX is meant to represent core processor X.
 - Gates appear as an n -entry byte-size array with read and write accesses ($n = 16, 32, 64$).
 - Processors lock gates by writing “processor_number+1” to the appropriate gate and must read back the gate value to verify the lock operation was successful.
 - Once locked, the gate is unlocked by a write of zeroes from the locking processor.
 - The number of implemented gates is specified by a hardware configuration define.
 - Each hardware gate appears as a 16-state, 4-bit state machine.
 - 16-state implementation
 - if gate = 0x0, then state = unlocked
 - if gate = 0x1, then state = locked by processor (master) 0
 - if gate = 0x2, then state = locked by processor (master) 1

- ...
- if gate = 0xF, then state = locked by processor (master) 14
- Uses the logical bus master number as a reference attribute plus the specified data patterns to validate all write operations.
 - Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor.
 - Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear_all capability.
 - Memory-mapped IPS slave peripheral platform module
 - Interface to the IPS bus for programming-model accesses

A simplified block diagram of the Semaphores module is shown in [Figure 192](#). In the diagram, the register blocks named gate0, gate1,..., gate ($n - 2$), gate ($n - 1$) include the finite state machines implementing the semaphore gates.

Figure 192. Semaphores2 block diagram



23.2 Memory map/register definition

Offset address	Register name	Access	Reset value	Section/ page
0x0000–0x003F	Semphores2 Gate n (GATE n)	R/W	0x00	23.2.1/704
0x0040	Semphores2 Reset Gate (RSTGT)	R/W	0x0000	23.2.2/705
0x0042–0x3FFF	Reserved			

23.2.1 Semaphores Gate n Register (SEMA4_GATE)

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value nor the appropriate lock value (processor_number + 1) are simply treated as “no operation” and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes.

Figure 193. Semaphore2 Gate Register (GATE)

Offset SEMA4_BASE + n ($n = 0, 1, 2, \dots, 15, \dots, 31, \dots, 63$)					Access: Supervisor Read/Write			
	0	1	2	3	4	5	6	7
R	0	0	0	0	GTFSM			
W								
Reset	0	0	0	0	0	0	0	0

Table 276. SEMA4_GATE field descriptions

Field	Description
0–3	Reserved
4–7 GTFSM	Gate Finite State Machine. The hardware gate is maintained in a 16-state implementation, defined as: 0000 The gate is unlocked (free). 0001 The gate has been locked by processor 0. 0010 The gate has been locked by processor 1. 0011 The gate has been locked by processor 2. 0100 The gate has been locked by processor 3. 0101 The gate has been locked by processor 4. 0110 The gate has been locked by processor 5. 0111 The gate has been locked by processor 6. 1000 The gate has been locked by processor 7. 1001 The gate has been locked by processor 8. 1010 The gate has been locked by processor 9. 1011 The gate has been locked by processor 10. 1100 The gate has been locked by processor 11. 1101 The gate has been locked by processor 12. 1110 The gate has been locked by processor 13. 1111 The gate has been locked by processor 14. <i>Note: The state of the gate reflects the last processor that locked it, which can be useful during system debug.</i> For more details, see Section 23.2.1, Semaphores Gate n Register (SEMA4_GATE).

23.2.2 Semaphores (Secure) Reset Gate (SEMA4_RSTGT)

Although the intent of the hardware gate implementation specifies a protocol where the locking processor must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a “secure” reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the specified gate(s). The required access pattern is:

1. A processor performs a 16-bit write to the SEMA4_RSTGT memory location. The most significant byte (SEMA4_RSTGT[RSTGDP]) must be 0xE2; the least significant byte is a “don’t_care” for this reference.
2. The same processor then performs a second 16-bit write to the SEMA4_RSTGT location. For this write, the upper byte (SEMA4_RSTGT[RSTGDP]) is the logical complement of the first data pattern (0x1D) and the lower byte (SEMA4_RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or else that all gates are to be cleared. If the same processor writes incorrect data on the second access or another processor performs the second write access, the special gate reset sequence is aborted and no error signal will be asserted.
3. Reads of the SEMA4_RSTGT location return information on the 2-bit state machine (SEMA4_RSTGT[RSTGSM]) that implements this function, the bus master performing the reset (SEMA4_RSTGT[RSTGMS]), and the gate number(s) last cleared

(SEMA4_RSTGT[RSTGTN]). Reads of the SEMA4_RSTGT register do not affect the secure reset finite state machine in any manner.

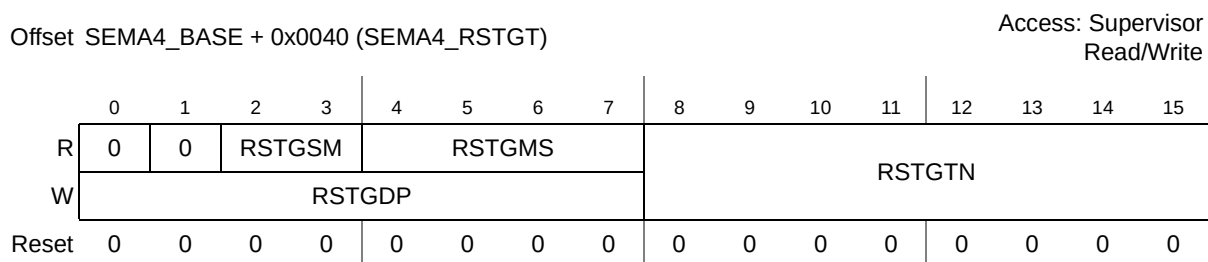


Figure 194. Semaphore2 Reset Gate Register (RSTGT)

Table 277. RSTGT field descriptions

Field	Description
0–1	Reserved
2–3 RSTGSM	Reset Gate Finite State Machine. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as: 00 Idle, waiting for the first data pattern write. 01 Waiting for the second data pattern write. 10 The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The “10” state persists for only one clock cycle. Software will never be able to observe this state. 11 This state encoding is never used and therefore reserved. Reads of the SEMA4_RSTGT register return the encoded state machine value. Note the RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value.
4–7 RSTGMS	Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information.
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then <i>reset the single gate</i> defined by RSTGTN, else <i>reset all the gates</i>.
0–7 RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = 0xE2 while the second write requires RSTGDP = 0x1D.

23.3 Functional description

Functional operation of the Semaphores module and specific details of the state machines of the SEMA4_GATE n registers are provided as follows.

As described previously, each of the SEMA4_GATE n registers implements a 4-bit, 16-state machine. A *simplified* diagram of the state transitions for each gate is shown in [Figure 195](#).

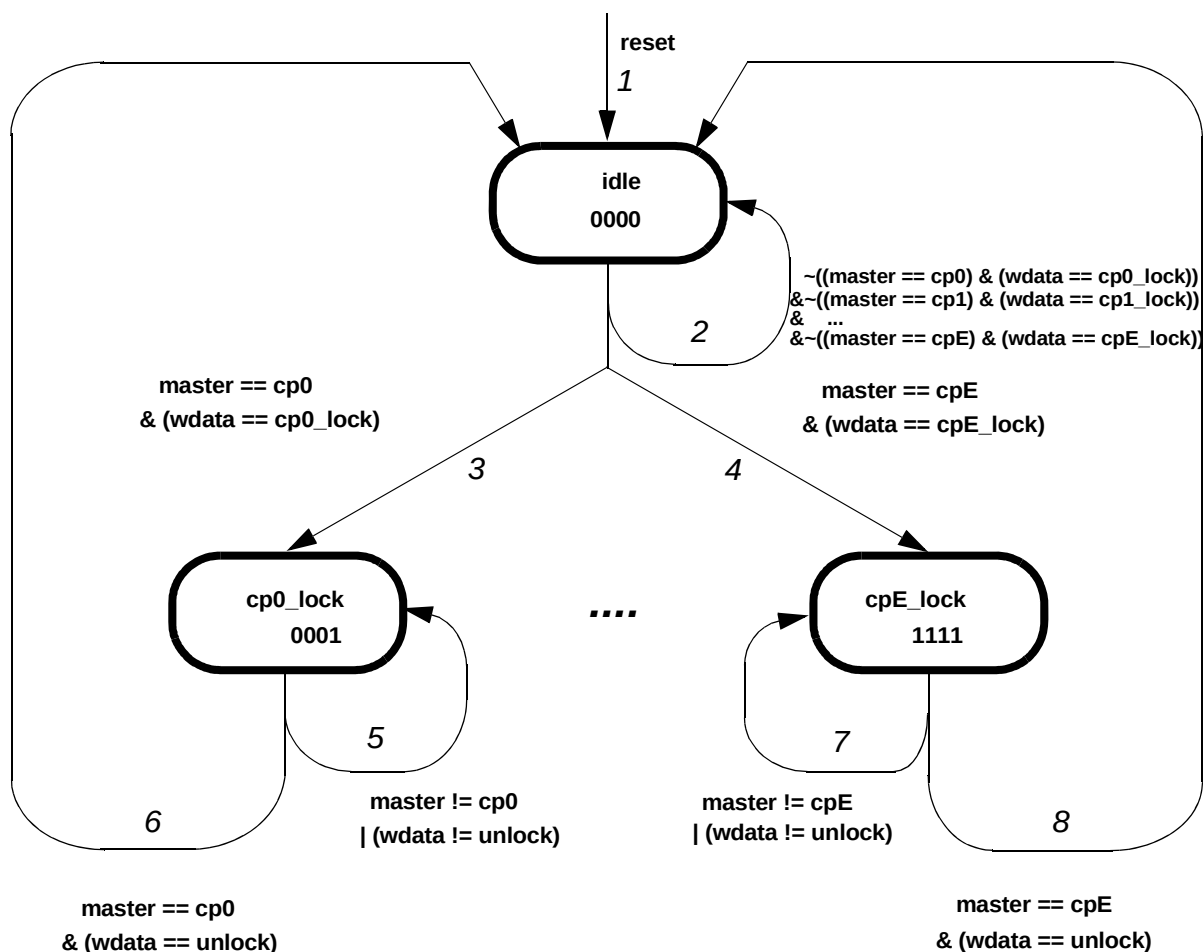


Figure 195. SEMA4_GATE n state machine

The bus master number is used to identify core processor X (cp X) where the notation cpE is used to represent core processor 14 (= 0xE). The platform passes the AHB bus master number through the AIPS controller and drives the signal to the Semaphores module as an IPS sideband signal.

The state transitions for SEMA4_GATE n are defined in [Table 278](#).

Table 278. SEMA4_GATE n state transitions

Current State	Next State	Transition	Description
—	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.

Table 278. SEMA4_GATE n state transitions (continued)

Current State	Next State	Transition	Description
idle	cp0_lock	3	When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state.
idle	cpE_lock	4	When a write of the "cpE_lock" value is initiated by processor 0xE, the gate transitions into the cpE_lock state.
cp0_lock	cp0_lock	5	Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs.
cpE_lock	cpE_lock	7	Once in this state, the gate remains here if any attempted write is not from cpE with the unlock data value.
cpE_lock	idle	8	The gate returns to the idle (unlocked) state once a write from cpE with the unlock data value occurs.

As previously noted, the following gate data values are used:

- The lock data value is (processor number + 1) where the processor number and the platform bus master number are the same.
- The unlock data value is 0x00.

24 Interrupt Controller (INTC)

24.1 Introduction

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or more processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and automotive powertrain applications where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 64 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

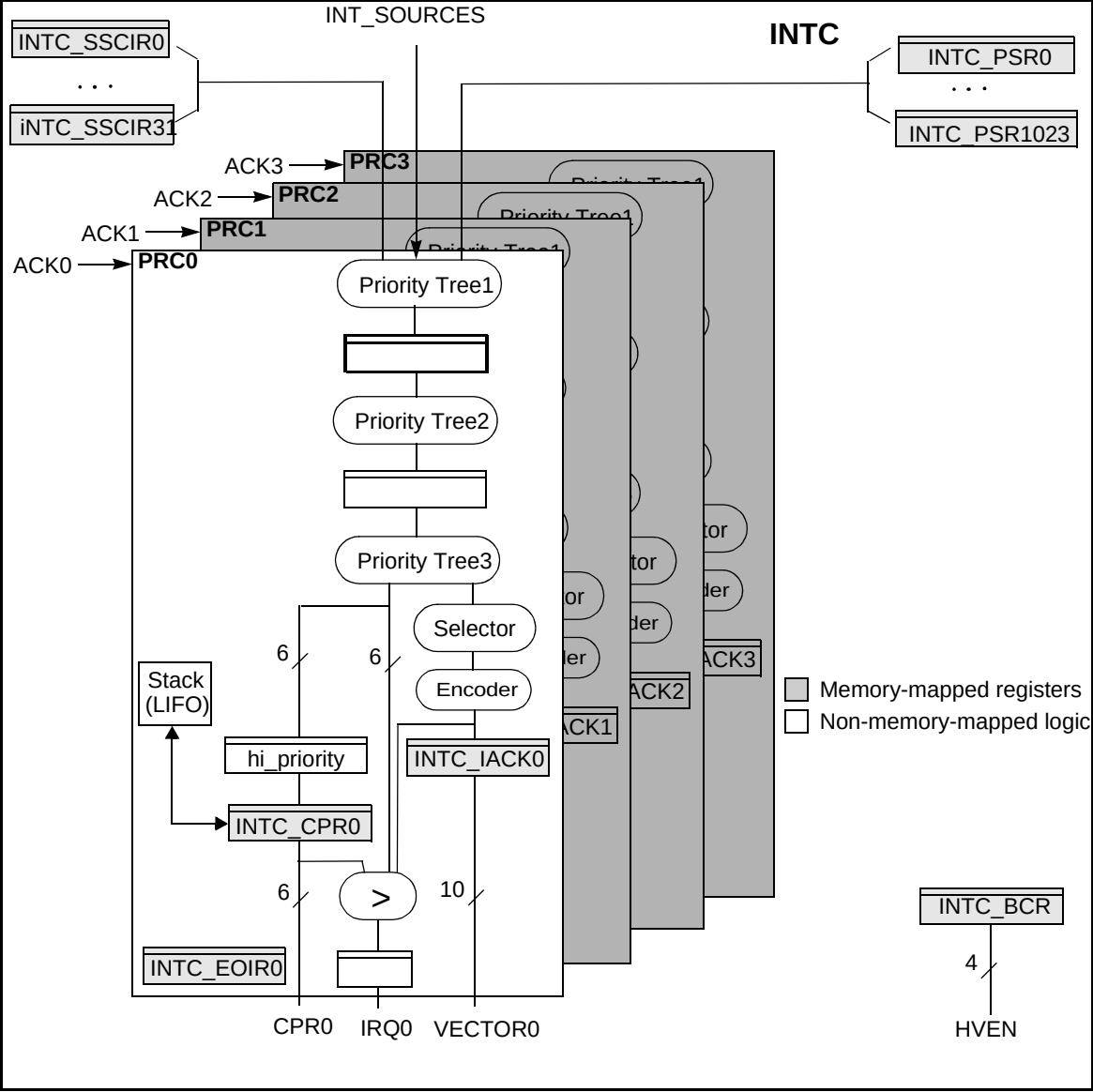
When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

24.2 Block diagram

The block diagram for the INTC is shown in [Figure 196](#).

Figure 196. INTC block diagram



24.3 Features

- Each peripheral interrupt source is software-steerable to processor 0, processor 1, processor 2, processor 3, or any combination of interrupt request outputs
- 32 software-settable interrupt request sources
- 10-bit vector
 - Unique vector for each interrupt request source
 - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 64 priorities
- Each interrupt source can be triggered by software
- Preemption
 - Preemptive prioritized interrupt requests to processor
 - ISR with higher priority preempts ISRs or tasks with lower priorities
 - Automatic pushing or popping of preempted priority to or from a LIFO
 - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
 - 3 INTC clock cycles from int request in to int request to CPU
- Low latency
 - 3 INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; four clock cycles from receipt of software request

24.4 Modes of operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 s of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 s for each IRQ).

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

24.4.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN_PRCn bit in the INTC_BCR (see [Section 24.5.2.1, INTC Block Configuration Register \(INTC_BCR\)](#)) is negated. The hardware vector enable signal to any processor is driven as negated when its associated HVEN_PRCn bit is negated. The vector is read from the INTC_IACKRn registers (see [Section 24.5.2.4, INTC Interrupt Acknowledge Register for Processor n \(INTC_IACKRn\)](#)). Reading the INTC_IACKRn negates the interrupt request to the associated processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in the INTC_CPRn (see [Section 24.5.2.3, INTC Current Priority Register for Processor n \(INTC_CPRn\)](#)) onto

the associated LIFO, and updates PRI in the associated INTC_CPR_PRC_n with the new priority.

24.4.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for a given processor when its associated HVEN_n bit in the INTC_BCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted.

When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKR_n, depending on which processor was assigned to handle a given interrupt source.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR_n register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR_n register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR_n does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs (see [Section 24.5.2.6, INTC Software Set/Clear Interrupt Registers \(INTC_SSCIR0_3–INTC_SSCIR28_31\)](#)) is written at a time such that the PRI value in the associated INTC_CPR_n register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR_n is updated with the new priority, and the associated LIFO is neither pushed or popped.

24.5 Memory map and register definition

24.5.1 Memory map

[Table 279](#) is the INTC memory map.

The actual availability and number of registers are chip-specific. For this information, see the chapter that describes how modules are configured and connected.

Table 279. INTC memory map

Address offset	Register	Location
000h	INTC Block Configuration Register (INTC_BCR)	on page 713
004h	INTC Master Protection Register (INTC_MPROT)	on page 713
008h–00Ch	Reserved	
010h	INTC Current Priority Register for Processor 0 (INTC_CPR0)	on page 715
014h	INTC Current Priority Register for Processor 1 (INTC_CPR1)	on page 715
018h	INTC Current Priority Register for Processor 2 (INTC_CPR2)	on page 715
01Ch	INTC Current Priority Register for Processor 3 (INTC_CPR3)	on page 715

Table 279. INTC memory map (continued)

Address offset	Register	Location
020h	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	on page 716
024h	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)	on page 716
028h	INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)	on page 716
02Ch	INTC Interrupt Acknowledge Register for Processor 3 (INTC_IACKR3)	on page 716
030h	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	on page 717
034h	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)	on page 717
038h	INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)	on page 717
03Ch	INTC End Of Interrupt Register for Processor 3 (INTC_EOIR3)	on page 717
040h–05Ch	INTC Software Set/Clear Interrupt Register 0–3 (INTC_SSCIR0_3)– INTC Software Set/Clear Interrupt Register 28–31 (INTC_SSCIR28_31)	on page 718
060h–85Eh	INTC Priority Select Registers (INTC_PSR0 – INTC_PSR1023)	on page 719
860h–FFC	Reserved	

Note: A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

24.5.2 Register descriptions

With the exception of the INTC_SSCIR n and INTC_PSR n registers, all of the registers are 32-bit. Any combination of accessing the four s of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two s, and aligned 32 bits.

Although INTC_SSI n are 8 bits wide and INTC_PSR n are 16 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR n) are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR n) does not affect the operation of the write.

24.5.2.1 INTC Block Configuration Register (INTC_BCR)

The Block Configuration Register is used to configure options of the INTC.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] has write access, otherwise, a termination error is asserted.

Figure 197. INTC Block Configuration Register (INTC_BCR)

Offset: 000h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	HVEN3	0	0	0	HVEN2	0	0	0	HVEN1	0	0	0	HVEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 280. INTC_BCR field descriptions

Field	Description
HVEN n	Hardware vector enable. Controls whether the INTC is in hardware vector mode or software vector mode. See Section 24.4, Modes of operation , for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

24.5.2.2 INTC Master Protection Register (INTC_MPROT)

The Master Protection Register is used to protect cores from writing to other cores INTC registers.

Figure 198. INTC Master Protection Register (INTC_MPROT)

Offset: 004h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ID	0	0	0	MPROT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 281. INTC_MPROT field descriptions

Field	Description
ID	ID This register can only be updated if MPROT=0 or if processor master ID=ID. When MPROT is set, the ID should also be updated to indicate which master can write to any register in the INTC memory map. 00 - master0 (reset default) 01 - master1 10 - master2 11 - master3
MPROT	Master Protection Controls what programming space a given master can write to. If MPROT is disabled, then all masters can write to any valid entry of the INTC memory map. If MPROT is enabled, INTC_MPROT[ID] can write to any valid entry of the INTC memory map but other masters can only write to their assigned resources. Once MPROT is enabled by a master, only that master may modify INTC_MPROT. 0 Master programming space protection disabled 1 Master programming space protection enabled

24.5.2.3 INTC Current Priority Register for Processor *n* (INTC_CPR_{*n*})

The INTC_CPR_{*n*} masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR_{*n*}[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR_{*n*}) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR_{*n*}) is written, the LIFO is popped into the INTC_CPR_{*n*} PRI field. An exception case in hardware vector mode to this behavior is described in [Section 24.4.2, Hardware vector mode](#).

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See [Section 24.7.5, Priority ceiling protocol](#).

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Note: *A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See [Section 24.7.5.2, Ensuring coherency](#), for example code to ensure coherency.*

Figure 199. INTC Current Priority Register for Processor *n* (INTC_CPR_{*n*})

Offsets: 010h (INTC_CPR0)
014h (INTC_CPR1)
018h (INTC_CPR2)
01Ch (INTC_CPR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PRI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Table 282. INTC_CPR_{*n*} field descriptions

Field	Description
PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

24.5.2.4 INTC Interrupt Acknowledge Register for Processor *n* (INTC_IACKR_{*n*})

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR_{*n*} has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR_{*n*} does not have side effects in hardware vector mode.

If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

Figure 200. INTC Interrupt Acknowledge Register for Processor *n* (INTC_IACKR_{*n*})

Offsets:	020h (INTC_IACKR0)	Access: User write always; non-speculative read ⁽¹⁾																	
	024h (INTC_IACKR1)																		
	028h (INTC_IACKR2)																		
	02Ch (INTC_IACKR3)																		
	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15																		
R	VTBA[19:4]																		
W																			
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																		
	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31																		
R	VTBA[3:0]				INTVEC												0	0	
W																			
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																		

1. When the corresponding INTC_BCR[HVEN_{*n*}] = 1, a read of the INTC_IACKR has no side effects.

Table 283. INTC_IACKR_{*n*} field descriptions

Field	Description
VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.

24.5.2.5 INTC End of Interrupt Register for Processor *n* (INTC_EOIR_{*n*})

Writing to the INTC_EOIR_{*n*} signals the end of the servicing of the interrupt request. When the INTC_EOIR_{*n*} is written, the priority last pushed on the LIFO is popped into INTC_CPR_{*n*}. The values and size of data written to the INTC_EOIR_{*n*} are ignored. Those values and sizes written to this register neither update the INTC_EOIR_{*n*} contents nor affect whether the LIFO pops. When you write to this register, write four all-zero s.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Figure 201. INTC End of Interrupt Register for Processor n (INTC_EOIR n)

Offsets:	030h (INTC_EOIR0)																Access: User write-only
	034h (INTC_EOIR1)																
	038h (INTC_EOIR2)																
	03Ch (INTC_EOIR3)																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W	EOI[31:16]																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W	EOI[15:0]																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 284. INTC_EOIR n field descriptions

Field	Description
EOI	End of Interrupt. Write four all-zero s to this field to signal the end of the servicing of an interrupt request.

24.5.2.6 INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0_3–INTC_SSCIR28_31)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET n will leave SET n unchanged at 0 but will set CLR n . Writing a 0 to SET n will have no effect. CLR n is the flag bit. Writing a 1 to CLR n will clear it. Writing a 0 to CLR n will have no effect. If a 1 is written to a pair of SET n and CLR n bits at the same time, CLR n is asserted, regardless of whether CLR n was asserted before the write.

[Figure 202](#) shows the structure and numbering scheme of the first INTC_SSCIR. The other registers follow the same numbering scheme.

Any processor will be allowed to write to any of the SET bits in the INTC_SSCIR n registers.

If INTC_MPROT[MPROT]=0 (disabled), all processors can write to the CLR bit.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor can write to the CLR bit and not the SET bit, otherwise, a termination error is asserted.

Note: *If INTC_MPROT[MPROT] is enabled, processors with master ID $\neq$ INTC_MPROT[ID] must only write the INTC_SSCIR n registers using a byte operation. A write of any other size will result in a termination error.*

Figure 202. INTC Software Set/Clear Interrupt Register 0–3 (INTC_SSICR0_3)

Offset: 040h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	CLR0	0	0	0	0	0	0	0	CLR1
W							SET0	CLR0							SET1	CLR1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	CLR2	0	0	0	0	0	0	0	CLR3
W							SET2	CLR2							SET3	CLR3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 285. INTC_SSICRn field descriptions

Field	Description
SETn	Set flag bits. Writing a 1 will set the corresponding CLRn bit. Writing a 0 will have no effect. Each SETn is always read as a 0.
CLRn	Clear flag bits. CLRn is the flag bit. Writing a 1 to CLRn will clear it provided that a 1 is not written simultaneously to its corresponding SETn bit. Writing a 0 to CLRn will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

24.5.2.7 INTC Priority Select Registers (INTC_PSR0 – INTC_PSR1023)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSRn is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32–1023, and their priorities are configured in INTC_PSR32–1023, respectively.

[Figure 203](#) and [Figure 204](#) show the structure and numbering scheme of the first and last PSRs, respectively. The other registers follow the same numbering scheme.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access to INTC_PSRn[PRC_SEL,SWT], otherwise, a termination error is asserted. When writing the INTC_PSRn[PRI], only the processor with master ID (0-3) = INTC_MPROT[ID] and the processor who's master ID (0-3) matches the setting of the INTC_PSRn[PRC_SEL] will be allowed to write the INTC_PSRn[PRI]. An attempt by any other processor to write the INTC_PSRn[PRI] will result in a termination error.

Note: *S.Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.*

Interrupts are not detected while the PSRs are being written.

If INTC_MPROT[MPROT] is enabled, the core with master ID (0-3) = INTC_MPROT[ID] will be able to write the INTC_PSRn with either , half-word or word operations. All other master IDs are restricted to operations to the INTC_PSRn[PRI] bits (i.e the odd addresses). Any other sized write operation, or a operation to the INTC_PSRn[PRC_SEL] (i.e. the even) will result in a termination error.

Figure 203. INTC Priority Select Register 0 (INTC_PSR0)

Offset:	060h											Access: User read/write					
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	PRC_SEL0				0	0	0	0	0	0	PRI0						
W																	
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 204. INTC Priority Select Register 1023 (INTC_PSR1023)

Offset:	85Eh												Access: User read/write							
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	PRC_SEL1023				0	0	0	SWT1023	0	0	PRI1023									
W																				
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Table 286. INTC_PSRn field descriptions

Field	Description
PRC_SELn	Processor select bits. See Table 287 . If an interrupt source is enabled, PRC_SELn[3:0] selects whether the interrupt request is to be sent to processor 0, processor 1, processor 2, processor 3, or any combination.
SWTn	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a '1' to SWTn sets the bit. On the interrupt acknowledge cycle, the SWTn bit is cleared. The SWT function works both in hardware and software vector mode. It is only cleared by the enabled Processor(s) defined in the PRC_SELn field of the PSR.
PRI _n	Priority select. Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

Table 287. Selected Processor for Interrupt Request

PRC_SEL n [3:0]	Meaning
0000	No interrupt request sent
0001	Interrupt request sent to processors 3
0010	Interrupt request sent to processors 2
0011	Interrupt request sent to processors 2, 3
0100	Interrupt request sent to processors 1
0101	Interrupt request sent to processors 1, 3
0110	Interrupt request sent to processors 1, 2
0111	Interrupt request sent to processors 1, 2, 3
1000	Interrupt request sent to processors 0
1001	Interrupt request sent to processors 0, 3
1010	Interrupt request sent to processors 0, 2
1011	Interrupt request sent to processors 0, 2, 3
1100	Interrupt request sent to processors 0, 1
1101	Interrupt request sent to processors 0, 1, 3
1110	Interrupt request sent to processors 0, 1, 2
1111	Interrupt request sent to processors 0, 1, 2, 3

24.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

24.6.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI n value in INTC_PSR n is higher than the PRI value in INTC_CPR n , negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in the associated INTC_CPR n is updated with the corresponding PRI n value in INTC_PSR n .

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

24.6.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

24.6.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR n) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET n bit in INTC_SSCIR n . This write sets the corresponding CLR n bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR n bit. Specific behavior includes the following:

- Writing a 1 to SET n leaves SET n unchanged at 0 but sets the flag bit (which is the CLR n bit).
- Writing a 0 to SET n has no effect.
- Writing a 1 to CLR n clears the flag (CLR n) bit.
- Writing a 0 to CLR n has no effect.
- If a 1 is written to a pair of SET n and CLR n bits at the same time, the flag (CLR n) is set, regardless of whether CLR n was asserted before the write.

The time from the write to the SET n bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

Any Processor n will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the Processor n which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSR n [PRC_SEL n] bits.

24.6.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–31 are assigned vectors 0–31, respectively. The peripheral interrupt requests are assigned vectors from 32 to a number as high as needed to cover all peripheral interrupt requests.

24.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI n and PRC_SEL n values set in INTC_PSR n . The result of that comparison also is compared to PRI in the associated INTC_CPR n . The results of those comparisons are used to manage the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

24.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in [Figure 196](#) are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a

unique vector for the preempting peripheral or software-settable interrupt request is generated for the associated INTC_IACKR n , and if in hardware vector mode, for the interrupt vector provided to the processor.

24.6.2.1.1 Priority tree

The priority tree for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated selector logic.

24.6.2.1.2 Selector

If only one interrupt request from the associated priority tree is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority tree are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

24.6.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

24.6.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated INTC_CPR n . If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR n , or the PRI value in the associated INTC_CPR n is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR n when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI n in INTC_PSR n are zero will not cause a preemption because their PRI n will not be higher than PRI in the associated INTC_CPR n .

Another function of the comparator is to signal an update of the INTC_IACKR n with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of the INTC_CPR n occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

24.6.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated INTC_CPR n . Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the associated INTC_CPR n does not need to be loaded from the associated INTC_CPR n and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the associated INTC_CPR n .

The PRI value in the associated INTC_CPR n is pushed onto the LIFO when the associated INTC_IACKR n is read in software vector mode or the interrupt acknowledge signal from the

associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated INTC_CPR n whenever the associated INTC_EOIR n is written. An exception case in hardware vector mode to this behavior is described in [Section 24.4.2, Hardware vector mode](#).

Although the INTC supports up to 64 priorities, an ISR executing with PRI in the INTC_CPR n equal to 63 will not be preempted. Therefore, the LIFO supports the stacking of 63 priorities. However, the LIFO is only 62 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 63 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

24.6.3 Handshaking with processor

24.6.3.1 Software vector mode handshaking

24.6.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to [Figure 205](#).

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in [Figure 205](#). The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated INTC_CPR n , it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR n is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in [Section 24.4.1, Software vector mode](#).

24.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR n must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated INTC_CPR n . Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in the associated INTC_CPR n is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

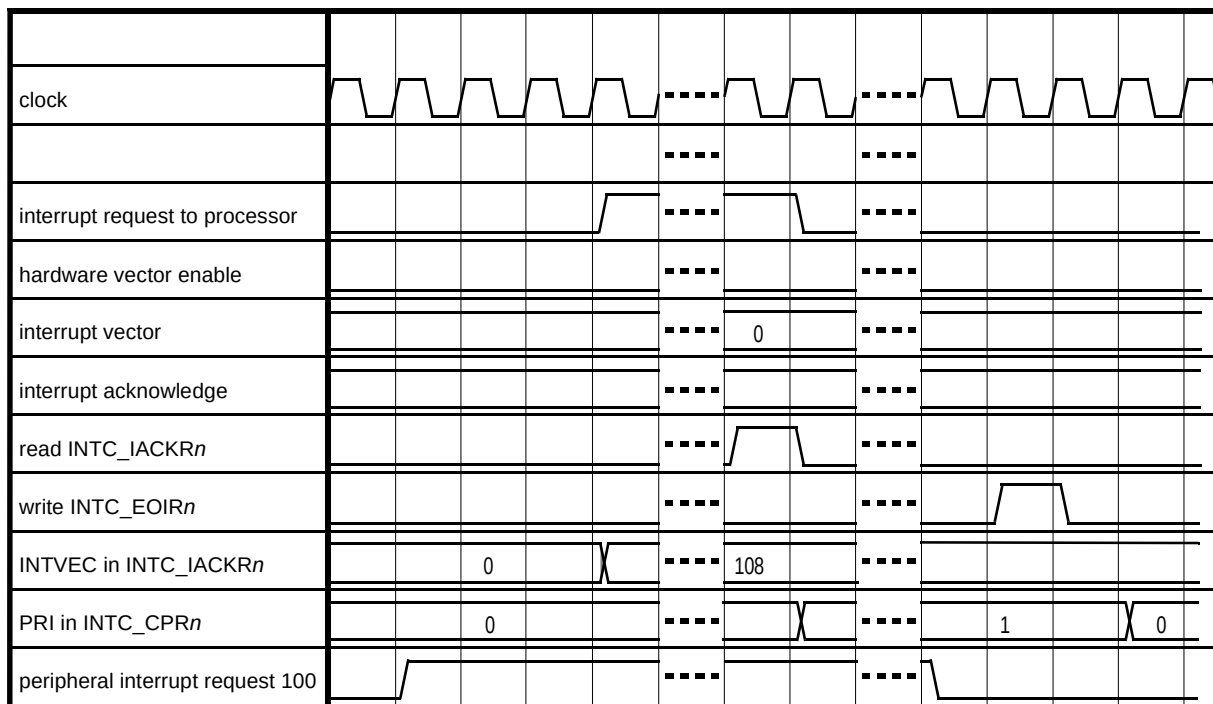


Figure 205. Timing diagram of software vector mode handshaking

24.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in [Figure 206](#). As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPRn, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKRn is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKRn. The rest of the handshaking is described in [Section 24.4.2, Hardware vector mode](#).

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIRn, is the same as in software vector mode. See [Section 24.6.3.1.2, End of interrupt exception handler](#).

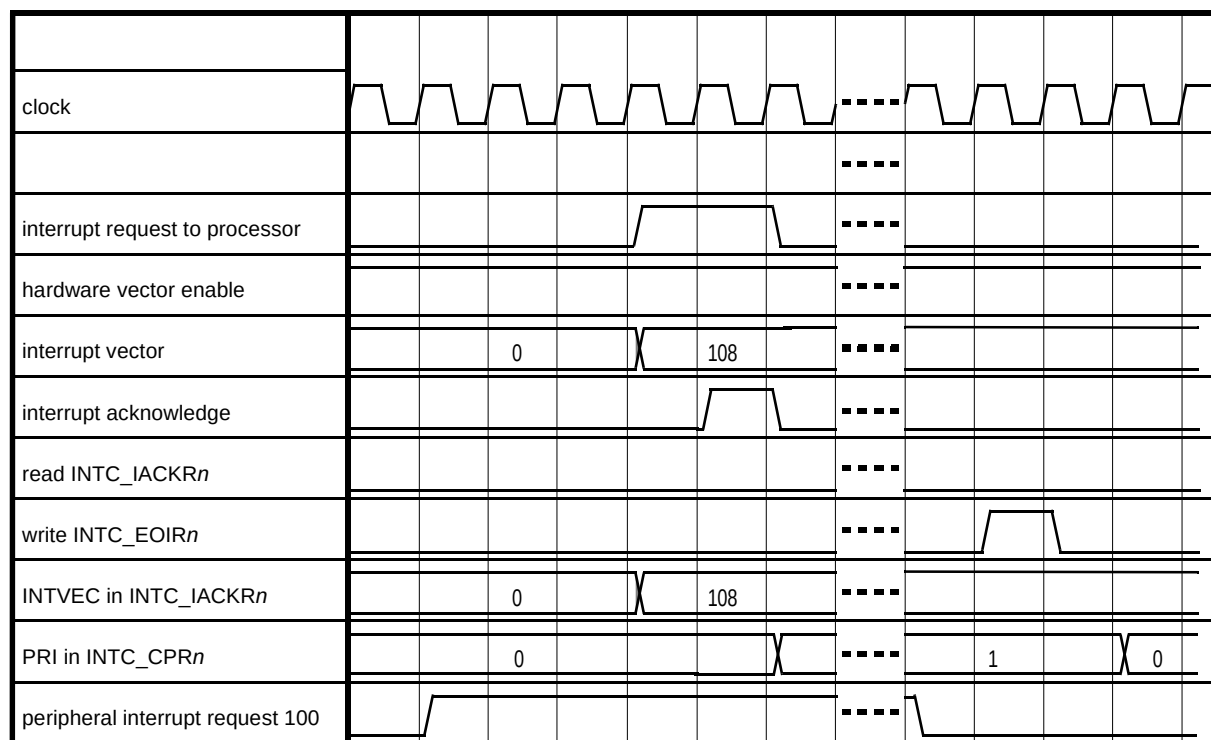


Figure 206. Timing diagram for hardware vector mode handshaking

24.7 Initialization/application information

24.7.1 Initialization flow

After exiting reset, all of the $PRIn$ and PRC_SELn fields in $INTC_PSRn$ are zero, and PRI in the $INTC_CPRn$ registers is 63. These reset values will prevent the INTC from asserting the interrupt request to the processors. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

```
interrupt_request_initialization:
configure HVENn in INTC_BCR
configure VTBAn in INTC_IACKRn
raise the  $PRIn$  fields and set the  $PRC\_SELn$  fields to the desired processor
in INTC_PSRn
set the enable bits or clear the mask bits for the peripheral interrupt
requests
lower  $PRI$  in INTC_CPRn to zero
enable processor(s) recognition of interrupts
```

24.7.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

24.7.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 s of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
```

```
b interrupt_exception_handler_continued# 16 s available, branch to
continue
```

```
interrupt_exception_handler_continued:
```

```
code to create stack frame, save working register, and save SRR0 and SRR1
(Power Architecture Save/Restore registers) (not shown)
```

```
lisr3,INTC_IACKRn@ha# form adjusted upper half of INTC_IACKRn address
```

```
lwzr3,INTC_IACKRn@l(r3)# load INTC_IACKRn, which clears request to
processor
```

```
lwzr3,0x0(r3)# load address of ISR from vector table
```

```
wrteei1 # enable processor recognition of interrupts
```

```
code to save rest of context required by Power Architecture EABI (Embedded
Binary Application Interface) (not shown)
```

```
mtlrr3 # move the INTC_IACKRn address into the link register
```

```
blrl # branch to ISR; link register updated with epilog
```

```
# address
```

```
epilog:
```

```
code to restore most of context required by Power Architecture EABI (not
shown)
```

```
# Popping the LIFO after the restoration of most of the context and the
disabling of processor
```

```
# recognition of interrupts eases the calculation of the maximum stack depth
at the cost of
```

```
# postponing the servicing of the next interrupt request.
```

```
mbar # ensure store to clear flag bit has completed
```

```
lisr3,INTC_EOIRn@ha# form adjusted upper half of INTC_EOIRn address
```

```
li r4,0x0 # form 0 to write to INTC_EOIRn
```

```
wrteei0 # disable processor recognition of interrupts
```

```
stwr4,INTC_EOIRn@l(r3) # store to INTC_EOIRn, informing INTC to lower
priority
```

```
code to restore SRR0 and SRR1, restore working registers, and delete stack
frame (not shown)
```

```
rfi
```

```
vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
.
.
.
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023
```

```
ISRn:
code to service the interrupt event (not shown)
code to clear flag bit which drives interrupt request to INTC (not shown)
```

```
blr      # return to epilog
```

24.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handler_n:
b interrupt_exception_handler_continuedn# 16 s available, branch to
continue
```

```
interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1
(not shown)
```

```
wrteei1  # enable processor recognition of interrupts
```

```
code to save rest of context required by Power Architecture EABI (not shown)
```

```
bl ISRn  # branch to ISR for interrupt with vector n
```

```
epilog:
code to restore most of context required by Power Architecture EABI (not
shown)
```

```

# Popping the LIFO after the restoration of most of the context and the
# disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth
# at the cost of
# postponing the servicing of the next interrupt request.
mbar          # ensure store to clear flag bit has completed
lisr3,INTC_EOIRn@ha# form adjusted upper half of INTC_EOIRn address
li r4,0x0     # form 0 to write to INTC_EOIRn
wrteei0       # disable processor recognition of interrupts
stwr4,INTC_EOIRn@l(r3)# store to INTC_EOIRn, informing INTC to lower
priority

```

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

```
rfi
```

ISRn:

code to service the interrupt event

code to clear flag bit which drives interrupt request to INTC

```
blr          # branch to epilog
```

24.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPRn having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPRn priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPRn priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPRn priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPRn while the shared resource is being accessed.

An ISR whose PRIn in INTC_PSRn has a value of 0 will not cause an interrupt request to the selected processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

24.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority

is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in [Table 288](#) shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 288. Order of ISR execution example

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 ⁽¹⁾	ISR208	ISR308	ISR408	interrupt exception handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIRn.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIRn.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIRn.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIRn.						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

24.7.5 Priority ceiling protocol

24.7.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR_n) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR_n to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR_n to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

24.7.5.2 Ensuring coherency

24.7.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR_n. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR_n can be made by those system services with the code sequence:

GetResource:

```
wrtteei 0# disable external interrupts to the Processor
raise PRI# Write to CPR, cache inhibit, guarded
mbar # flush out writes from store buffer
wrtteei 1# enable external interrupts to the Processor
isync# re-fetch Processor pipeline
```

ReleaseResource:

```
mbar # flush out writes from store buffer
wrtteei 0# disable external interrupts to the Processor
lower PRI# Write to CPR, cache inhibit, guarded
```



```
wrtteei 1# enable external interrupts to the Processor
```

Figure 207. OSEK Code Sequence

24.7.5.2.1.1 Interrupt request to processor

Referencing [Figure 196](#), the interrupt request logic to the processor is shown in [Figure 208](#).

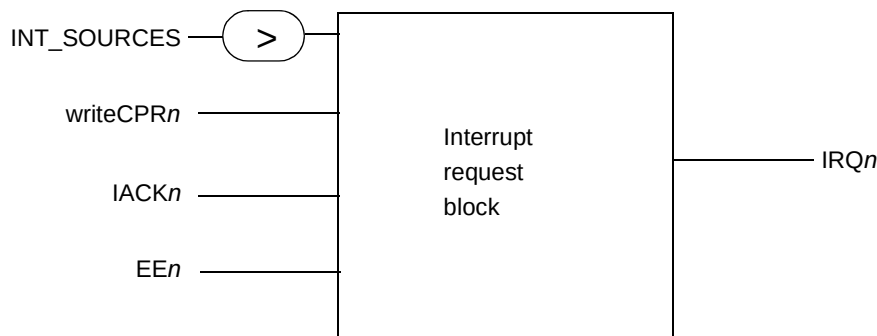


Figure 208. Interrupt request block diagram

Assuming that there are no $IRQn$ deasserted (no pending interrupt requests to processor n), if one of the external interrupt sources ($INT_SOURCES$) has a priority higher than the $INTC_CPRn$, then interrupt request to the processor ($IRQn$) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge ($IACK$) is asserted
- If the $EEEn$ bit has been cleared by the `wrtteei` instruction (see [Figure 207](#)), $IRQn$ will be re-evaluated when processor n writes to the $INTC_CPRn$ while processor n ignores $IRQn$

This provides a safe way to guarantee that no interrupts will be recognized by processor n while updating the $INTC_CPRn$.

24.7.5.2.2 Raised priority preserved

Before the instruction after the `GetResource` system service executes, all pending transactions have completed. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. Also, during the epilog of the interrupt exception handler for this preempting ISR, the raised priority has been restored from the LIFO to PRI in $INTC_CPRn$. The shared coherent data block now can be accessed coherently. [Figure 209](#) shows the timing diagram for this scenario, and [Table 289](#) explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

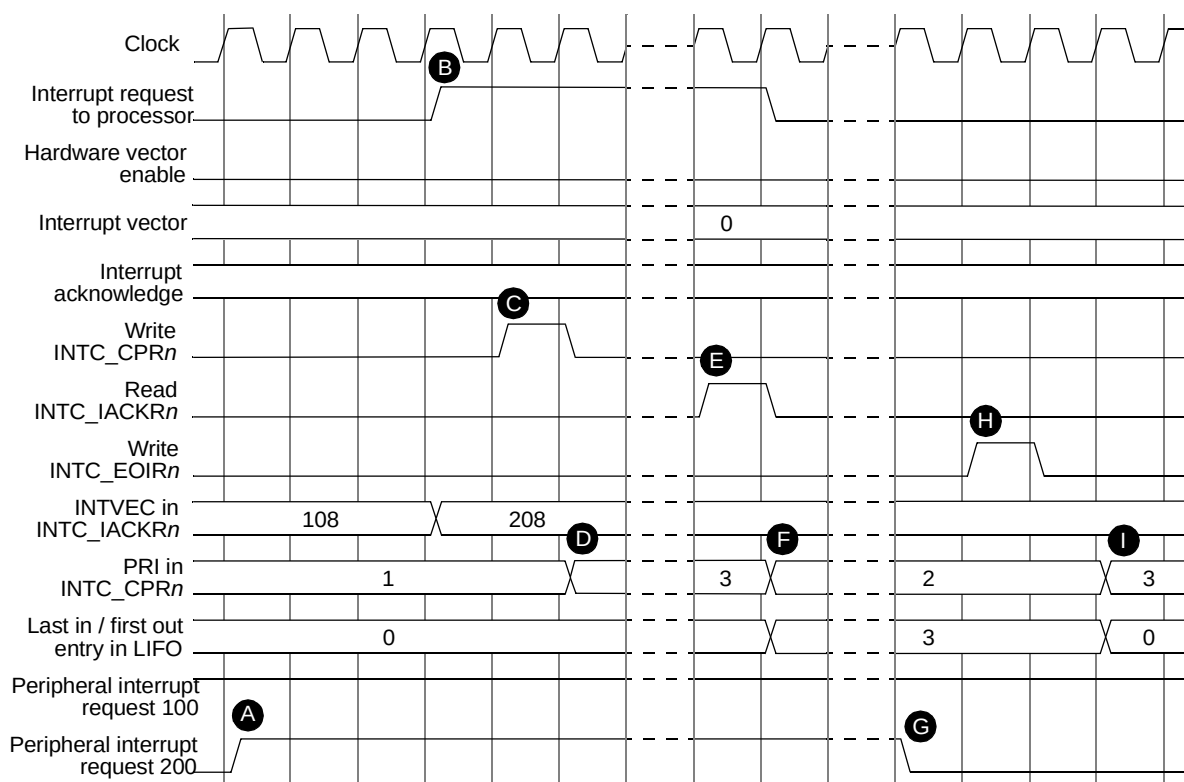


Figure 209. Timing diagram of raised priority preserved

Table 289. Raised priority preserved events

Event	Description
A	Peripheral interrupt request 200 asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKRn updates with vector for that peripheral interrupt request.
C	ISR108 writes to INTC_CPRn to raise priority to 3 before accessing shared coherent data block.
D	PRI in INTC_CPRn now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
E	Interrupt exception handler prolog acknowledges interrupt by reading INTC_IACKRn.
F	PRI of 3 pushed onto LIFO. PRI in INTC_CPRn updates to 2, the priority of ISR208.
G	ISR208 clears its flag bit, deasserting its peripheral interrupt request.
H	Interrupt exception handler epilog writes to INTC_EOIRn.
I	LIFO pops 3, restoring the raised priority onto PRI in INTC_CPRn. Next value to pop from LIFO is the priority from before peripheral interrupt request 100 interrupted. ISR108 now can access data block coherently after interrupt exception handler executes rfi instruction.

24.7.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 μ s, ISR2 executes every 200 μ s, and ISR3 executes every 300 μ s. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 μ s, then it has a higher priority than ISR2.

The INTC supports 64 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 μ s would share a priority, ISRs with request rates around 250 μ s would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

24.7.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

24.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRI_n value in $INTC_PSR_n$, which becomes the PRI value in $INTC_CPR_n$ with the interrupt acknowledge. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET_n bit in $INTC_SSCIR_n$. Writing a 1 to SET_n causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI_n value in the $INTC_PSR_n$, therefore will not cause preemptive scheduling inefficiencies.

24.7.7.2 Scheduling an ISR on another processor

Since the SET n bits in the INTC_SSCIR n are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR n bit in INTC_SSCIR n is asserted before again writing a 1 to the SET n bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET n bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR n bit and then writes 1 to a SET n bit on the first processor, informing it that it now can access the block of data.

24.7.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR n , one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in [Section 24.7.7.1, Scheduling a lower priority portion of an ISR](#)) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note: Lowering the PRI value in INTC_CPR n within an ISR to below the ISR's corresponding PRI value in INTC_PSR n allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

24.7.9 Negating an interrupt request outside of its ISR

24.7.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

24.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

24.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI_n values in $INTC_PSR_n$ must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to $INTC_SSCIR_n$ as the clearing of the flag bit that caused the present ISR to be executed. Refer to [Section 24.6.3.1.2, End of interrupt exception handler](#) for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI_n value in $INTC_PSR_n$.

24.7.10 Examining LIFO contents

Normally the user does not need to know the contents of the LIFO, or even how deep the LIFO is nested. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register ($INTC_CPR_n$). Disabling processor recognition of interrupts while examining the LIFO contents provides a coherent view of the preempted priorities. The code sequence is:

```
:
pop_lifo:
    store to INTC_EOIRn
    load INTC_CPRn, examine PRI, and store onto stack
    if PRI is not zero or value when interrupts were enabled, branch to
pop_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
    load stacked PRI value and store to INTC_CPRn
    load INTC_IACKRn
    if stacked PRI values are not depleted, branch to push_lifo
```

24.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chapter that describes how modules are configured and connected.

25 Enhanced Direct Memory Access (eDMA)

25.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes a DMA engine that performs source- and destination-address calculations, and the actual data-movement operations, along with local memory containing transfer control descriptors for each of the 64 channels.

Figure 210 illustrates the eDMA module.

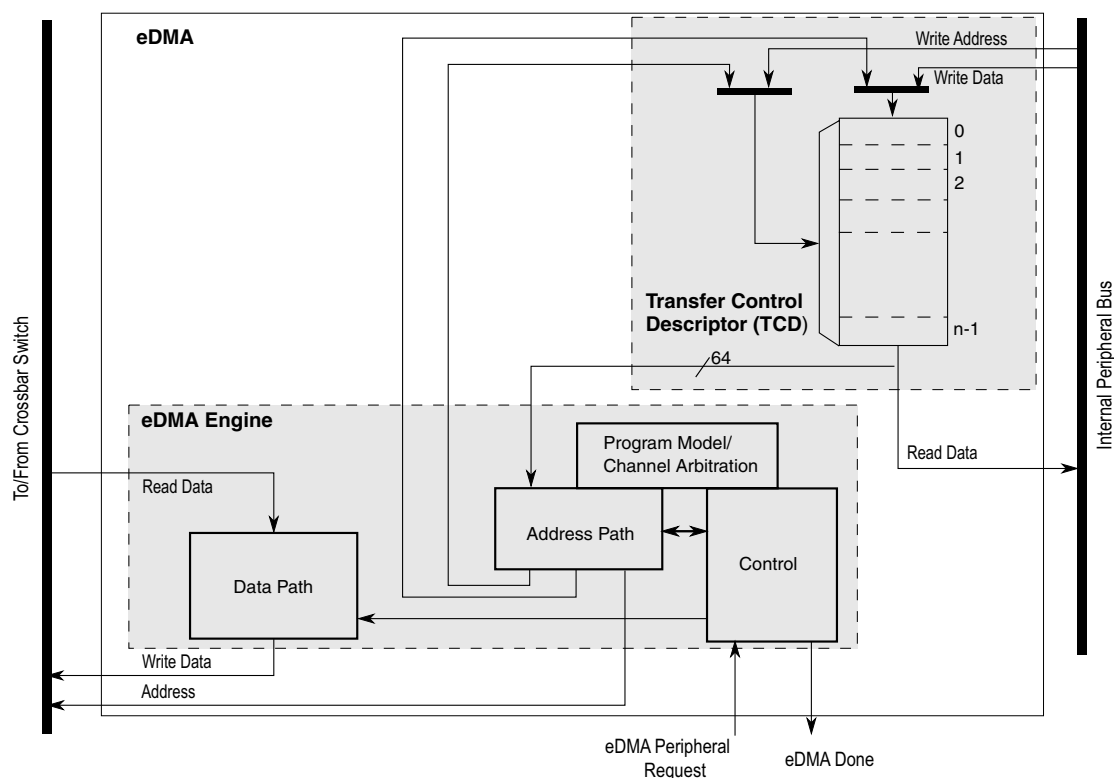


Figure 210. eDMA block diagram

25.1.1 Features

The eDMA is a highly-programmable data-transfer engine optimized to minimize the intervention from the host processor. It is intended for use in applications where the data

size to be transferred is statically known and not defined within the data packet itself. The eDMA module features:

- All data movement via dual-address transfers—read from source, write to destination: Programmable source and destination addresses and transfer size, plus support for enhanced addressing modes
- 64-channel implementation that performs complex data transfers with minimal intervention from a host processor
 - Internal data buffer, used as temporary storage to support 16- and 32-byte burst transfers
 - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
 - 32-byte TCD stored in local memory for each channel
 - An inner data transfer loop defined by a minor byte transfer count
 - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
 - Explicit software initiation
 - Initiation via a channel-to-channel linking mechanism for continuous transfers
 - Peripheral-paced hardware requests (one per channel)
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
 - One interrupt per channel, optionally asserted at completion of major iteration count
 - Optional error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Optional support for scatter/gather DMA processing
- Support for complex data structures
- Support to cancel transfers via software

Throughout this chapter, *n* is used to reference the channel number.

25.2 Modes of operation

The eDMA supports the following modes of operation.

25.2.1 Normal mode

In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.

A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.

25.2.2 Debug mode

DMA operation is configurable in Debug mode via the control register:

- If CR[EDBG] is cleared, the DMA continues to operate.
- If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.

25.2.3 Wait mode

Before entering Wait mode the DMA attempts to complete its current transfer. After the transfer completes the device enters Wait mode.

25.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions.
- The second region corresponds to the local transfer control descriptor memory.

Some registers are implemented as two 32-bit registers, and include an H and L suffix, signaling the high and low portions of the control function. [Table 290](#) provides a 32-bit view of the eDMA's memory map.

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0 to channel 63. Each TCD_n definition is presented as 11 registers of 16 or 32 bits.

Reading reserved bits in a register returns the value of zero and writes to reserved bits in a register are ignored. Reading or writing a reserved memory location generates a bus error.

Table 290. DMA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_0000	Control Register (DMA_CR)	32	R/W	0000_0000h	on page 783
FC0A_0004	Error Status Register (DMA_ES)	32	R	0000_0000h	on page 785
FC0A_0008	Enable Request Register High (DMA_ERQH)	32	R/W	0000_0000h	on page 787
FC0A_000C	Enable Request Register Low (DMA_ERQL)	32	R/W	0000_0000h	on page 790
FC0A_0010	Enable Error Interrupt Register High (DMA_EEIH)	32	R/W	0000_0000h	on page 793
FC0A_0014	Enable Error Interrupt Register Low (DMA_EEIL)	32	R/W	0000_0000h	on page 796
FC0A_0018	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	on page 799
FC0A_0019	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	on page 800

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_001A	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	on page 801
FC0A_001B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	on page 801
FC0A_001C	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	on page 802
FC0A_001D	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	on page 803
FC0A_001E	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	on page 803
FC0A_001F	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	on page 804
FC0A_0020	Interrupt Request Register High (DMA_INTH)	32	R/W	0000_0000h	on page 805
FC0A_0024	Interrupt Request Register Low (DMA_INTL)	32	R/W	0000_0000h	on page 808
FC0A_0028	Error Register High (DMA_ERRH)	32	R/W	0000_0000h	on page 811
FC0A_002C	Error Register Low (DMA_ERRL)	32	R/W	0000_0000h	on page 814
FC0A_0030	Hardware Request Status Register High (DMA_HRSH)	32	R/W	0000_0000h	on page 817
FC0A_0034	Hardware Request Status Register Low (DMA_HRSL)	32	R/W	0000_0000h	on page 820
FC0A_0100	Channel 0 Priority Register (DMA_DCHPRI0)	8	R/W	0Xh	on page 823
FC0A_0101	Channel 1 Priority Register (DMA_DCHPRI1)	8	R/W	0Xh	on page 823
FC0A_0102	Channel 2 Priority Register (DMA_DCHPRI2)	8	R/W	0Xh	on page 823
FC0A_0103	Channel 3 Priority Register (DMA_DCHPRI3)	8	R/W	0Xh	on page 823
FC0A_0104	Channel 4 Priority Register (DMA_DCHPRI4)	8	R/W	0Xh	on page 823
FC0A_0105	Channel 5 Priority Register (DMA_DCHPRI5)	8	R/W	0Xh	on page 823
FC0A_0106	Channel 6 Priority Register (DMA_DCHPRI6)	8	R/W	0Xh	on page 823
FC0A_0107	Channel 7 Priority Register (DMA_DCHPRI7)	8	R/W	0Xh	on page 823
FC0A_0108	Channel 8 Priority Register (DMA_DCHPRI8)	8	R/W	0Xh	on page 823

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_0109	Channel 9 Priority Register (DMA_DCHPRI9)	8	R/W	0Xh	on page 823
FC0A_010A	Channel 10 Priority Register (DMA_DCHPRI10)	8	R/W	0Xh	on page 823
FC0A_010B	Channel 11 Priority Register (DMA_DCHPRI11)	8	R/W	0Xh	on page 823
FC0A_010C	Channel 12 Priority Register (DMA_DCHPRI12)	8	R/W	0Xh	on page 823
FC0A_010D	Channel 13 Priority Register (DMA_DCHPRI13)	8	R/W	0Xh	on page 823
FC0A_010E	Channel 14 Priority Register (DMA_DCHPRI14)	8	R/W	0Xh	on page 823
FC0A_010F	Channel 15 Priority Register (DMA_DCHPRI15)	8	R/W	0Xh	on page 823
FC0A_0110	Channel 16 Priority Register (DMA_DCHPRI16)	8	R/W	0Xh	on page 823
FC0A_0111	Channel 17 Priority Register (DMA_DCHPRI17)	8	R/W	0Xh	on page 823
FC0A_0112	Channel 18 Priority Register (DMA_DCHPRI18)	8	R/W	0Xh	on page 823
FC0A_0113	Channel 19 Priority Register (DMA_DCHPRI19)	8	R/W	0Xh	on page 823
FC0A_0114	Channel 20 Priority Register (DMA_DCHPRI20)	8	R/W	0Xh	on page 823
FC0A_0115	Channel 21 Priority Register (DMA_DCHPRI21)	8	R/W	0Xh	on page 823
FC0A_0116	Channel 22 Priority Register (DMA_DCHPRI22)	8	R/W	0Xh	on page 823
FC0A_0117	Channel 23 Priority Register (DMA_DCHPRI23)	8	R/W	0Xh	on page 823
FC0A_0118	Channel 24 Priority Register (DMA_DCHPRI24)	8	R/W	0Xh	on page 823
FC0A_0119	Channel 25 Priority Register (DMA_DCHPRI25)	8	R/W	0Xh	on page 823
FC0A_011A	Channel 26 Priority Register (DMA_DCHPRI26)	8	R/W	0Xh	on page 823
FC0A_011B	Channel 27 Priority Register (DMA_DCHPRI27)	8	R/W	0Xh	on page 823
FC0A_011C	Channel 28 Priority Register (DMA_DCHPRI28)	8	R/W	0Xh	on page 823

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_011D	Channel 29 Priority Register (DMA_DCHPRI29)	8	R/W	0Xh	on page 823
FC0A_011E	Channel 30 Priority Register (DMA_DCHPRI30)	8	R/W	0Xh	on page 823
FC0A_011F	Channel 31 Priority Register (DMA_DCHPRI31)	8	R/W	0Xh	on page 823
FC0A_0120	Channel 32 Priority Register (DMA_DCHPRI32)	8	R/W	0Xh	on page 823
FC0A_0121	Channel 33 Priority Register (DMA_DCHPRI33)	8	R/W	0Xh	on page 823
FC0A_0122	Channel 34 Priority Register (DMA_DCHPRI34)	8	R/W	0Xh	on page 823
FC0A_0123	Channel 35 Priority Register (DMA_DCHPRI35)	8	R/W	0Xh	on page 823
FC0A_0124	Channel 36 Priority Register (DMA_DCHPRI36)	8	R/W	0Xh	on page 823
FC0A_0125	Channel 37 Priority Register (DMA_DCHPRI37)	8	R/W	0Xh	on page 823
FC0A_0126	Channel 38 Priority Register (DMA_DCHPRI38)	8	R/W	0Xh	on page 823
FC0A_0127	Channel 39 Priority Register (DMA_DCHPRI39)	8	R/W	0Xh	on page 823
FC0A_0128	Channel 40 Priority Register (DMA_DCHPRI40)	8	R/W	0Xh	on page 823
FC0A_0129	Channel 41 Priority Register (DMA_DCHPRI41)	8	R/W	0Xh	on page 823
FC0A_012A	Channel 42 Priority Register (DMA_DCHPRI42)	8	R/W	0Xh	on page 823
FC0A_012B	Channel 43 Priority Register (DMA_DCHPRI43)	8	R/W	0Xh	on page 823
FC0A_012C	Channel 44 Priority Register (DMA_DCHPRI44)	8	R/W	0Xh	on page 823
FC0A_012D	Channel 45 Priority Register (DMA_DCHPRI45)	8	R/W	0Xh	on page 823
FC0A_012E	Channel 46 Priority Register (DMA_DCHPRI46)	8	R/W	0Xh	on page 823
FC0A_012F	Channel 47 Priority Register (DMA_DCHPRI47)	8	R/W	0Xh	on page 823
FC0A_0130	Channel 48 Priority Register (DMA_DCHPRI48)	8	R/W	0Xh	on page 823

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_0131	Channel 49 Priority Register (DMA_DCHPRI49)	8	R/W	0Xh	on page 823
FC0A_0132	Channel 50 Priority Register (DMA_DCHPRI50)	8	R/W	0Xh	on page 823
FC0A_0133	Channel 51 Priority Register (DMA_DCHPRI51)	8	R/W	0Xh	on page 823
FC0A_0134	Channel 52 Priority Register (DMA_DCHPRI52)	8	R/W	0Xh	on page 823
FC0A_0135	Channel 53 Priority Register (DMA_DCHPRI53)	8	R/W	0Xh	on page 823
FC0A_0136	Channel 54 Priority Register (DMA_DCHPRI54)	8	R/W	0Xh	on page 823
FC0A_0137	Channel 55 Priority Register (DMA_DCHPRI55)	8	R/W	0Xh	on page 823
FC0A_0138	Channel 56 Priority Register (DMA_DCHPRI56)	8	R/W	0Xh	on page 823
FC0A_0139	Channel 57 Priority Register (DMA_DCHPRI57)	8	R/W	0Xh	on page 823
FC0A_013A	Channel 58 Priority Register (DMA_DCHPRI58)	8	R/W	0Xh	on page 823
FC0A_013B	Channel 59 Priority Register (DMA_DCHPRI59)	8	R/W	0Xh	on page 823
FC0A_013C	Channel 60 Priority Register (DMA_DCHPRI60)	8	R/W	0Xh	on page 823
FC0A_013D	Channel 61 Priority Register (DMA_DCHPRI61)	8	R/W	0Xh	on page 823
FC0A_013E	Channel 62 Priority Register (DMA_DCHPRI62)	8	R/W	0Xh	on page 823
FC0A_013F	Channel 63 Priority Register (DMA_DCHPRI63)	8	R/W	0Xh	on page 824
FC0A_0140	Channel 0 Master ID Register (DMA_DCHMID0)	8	R/W	00h	on page 824
FC0A_0141	Channel 1 Master ID Register (DMA_DCHMID1)	8	R/W	00h	on page 824
FC0A_0142	Channel 2 Master ID Register (DMA_DCHMID2)	8	R/W	00h	on page 824
FC0A_0143	Channel 3 Master ID Register (DMA_DCHMID3)	8	R/W	00h	on page 824
FC0A_0144	Channel 4 Master ID Register (DMA_DCHMID4)	8	R/W	00h	on page 824

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_0145	Channel 5 Master ID Register (DMA_DCHMID5)	8	R/W	00h	on page 824
FC0A_0146	Channel 6 Master ID Register (DMA_DCHMID6)	8	R/W	00h	on page 824
FC0A_0147	Channel 7 Master ID Register (DMA_DCHMID7)	8	R/W	00h	on page 824
FC0A_0148	Channel 8 Master ID Register (DMA_DCHMID8)	8	R/W	00h	on page 824
FC0A_0149	Channel 9 Master ID Register (DMA_DCHMID9)	8	R/W	00h	on page 824
FC0A_014A	Channel 10 Master ID Register (DMA_DCHMID10)	8	R/W	00h	on page 824
FC0A_014B	Channel 11 Master ID Register (DMA_DCHMID11)	8	R/W	00h	on page 824
FC0A_014C	Channel 12 Master ID Register (DMA_DCHMID12)	8	R/W	00h	on page 824
FC0A_014D	Channel 13 Master ID Register (DMA_DCHMID13)	8	R/W	00h	on page 824
FC0A_014E	Channel 14 Master ID Register (DMA_DCHMID14)	8	R/W	00h	on page 824
FC0A_014F	Channel 15 Master ID Register (DMA_DCHMID15)	8	R/W	00h	on page 824
FC0A_0150	Channel 16 Master ID Register (DMA_DCHMID16)	8	R/W	00h	on page 824
FC0A_0151	Channel 17 Master ID Register (DMA_DCHMID17)	8	R/W	00h	on page 824
FC0A_0152	Channel 18 Master ID Register (DMA_DCHMID18)	8	R/W	00h	on page 824
FC0A_0153	Channel 19 Master ID Register (DMA_DCHMID19)	8	R/W	00h	on page 824
FC0A_0154	Channel 20 Master ID Register (DMA_DCHMID20)	8	R/W	00h	on page 824
FC0A_0155	Channel 21 Master ID Register (DMA_DCHMID21)	8	R/W	00h	on page 824
FC0A_0156	Channel 22 Master ID Register (DMA_DCHMID22)	8	R/W	00h	on page 824
FC0A_0157	Channel 23 Master ID Register (DMA_DCHMID23)	8	R/W	00h	on page 824
FC0A_0158	Channel 24 Master ID Register (DMA_DCHMID24)	8	R/W	00h	on page 824

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_0159	Channel 25 Master ID Register (DMA_DCHMID25)	8	R/W	00h	on page 824
FC0A_015A	Channel 26 Master ID Register (DMA_DCHMID26)	8	R/W	00h	on page 824
FC0A_015B	Channel 27 Master ID Register (DMA_DCHMID27)	8	R/W	00h	on page 824
FC0A_015C	Channel 28 Master ID Register (DMA_DCHMID28)	8	R/W	00h	on page 824
FC0A_015D	Channel 29 Master ID Register (DMA_DCHMID29)	8	R/W	00h	on page 824
FC0A_015E	Channel 30 Master ID Register (DMA_DCHMID30)	8	R/W	00h	on page 824
FC0A_015F	Channel 31 Master ID Register (DMA_DCHMID31)	8	R/W	00h	on page 824
FC0A_0160	Channel 32 Master ID Register (DMA_DCHMID32)	8	R/W	00h	on page 824
FC0A_0161	Channel 33 Master ID Register (DMA_DCHMID33)	8	R/W	00h	on page 824
FC0A_0162	Channel 34 Master ID Register (DMA_DCHMID34)	8	R/W	00h	on page 824
FC0A_0163	Channel 35 Master ID Register (DMA_DCHMID35)	8	R/W	00h	on page 824
FC0A_0164	Channel 36 Master ID Register (DMA_DCHMID36)	8	R/W	00h	on page 824
FC0A_0165	Channel 37 Master ID Register (DMA_DCHMID37)	8	R/W	00h	on page 824
FC0A_0166	Channel 38 Master ID Register (DMA_DCHMID38)	8	R/W	00h	on page 824
FC0A_0167	Channel 39 Master ID Register (DMA_DCHMID39)	8	R/W	00h	on page 824
FC0A_0168	Channel 40 Master ID Register (DMA_DCHMID40)	8	R/W	00h	on page 824
FC0A_0169	Channel 41 Master ID Register (DMA_DCHMID41)	8	R/W	00h	on page 824
FC0A_016A	Channel 42 Master ID Register (DMA_DCHMID42)	8	R/W	00h	on page 824
FC0A_016B	Channel 43 Master ID Register (DMA_DCHMID43)	8	R/W	00h	on page 824
FC0A_016C	Channel 44 Master ID Register (DMA_DCHMID44)	8	R/W	00h	on page 824

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_016D	Channel 45 Master ID Register (DMA_DCHMID45)	8	R/W	00h	on page 824
FC0A_016E	Channel 46 Master ID Register (DMA_DCHMID46)	8	R/W	00h	on page 824
FC0A_016F	Channel 47 Master ID Register (DMA_DCHMID47)	8	R/W	00h	on page 824
FC0A_0170	Channel 48 Master ID Register (DMA_DCHMID48)	8	R/W	00h	on page 824
FC0A_0171	Channel 49 Master ID Register (DMA_DCHMID49)	8	R/W	00h	on page 824
FC0A_0172	Channel 50 Master ID Register (DMA_DCHMID50)	8	R/W	00h	on page 824
FC0A_0173	Channel 51 Master ID Register (DMA_DCHMID51)	8	R/W	00h	on page 824
FC0A_0174	Channel 52 Master ID Register (DMA_DCHMID52)	8	R/W	00h	on page 824
FC0A_0175	Channel 53 Master ID Register (DMA_DCHMID53)	8	R/W	00h	on page 824
FC0A_0176	Channel 54 Master ID Register (DMA_DCHMID54)	8	R/W	00h	on page 824
FC0A_0177	Channel 55 Master ID Register (DMA_DCHMID55)	8	R/W	00h	on page 824
FC0A_0178	Channel 56 Master ID Register (DMA_DCHMID56)	8	R/W	00h	on page 824
FC0A_0179	Channel 57 Master ID Register (DMA_DCHMID57)	8	R/W	00h	on page 824
FC0A_017A	Channel 58 Master ID Register (DMA_DCHMID58)	8	R/W	00h	on page 824
FC0A_017B	Channel 59 Master ID Register (DMA_DCHMID59)	8	R/W	00h	on page 824
FC0A_017C	Channel 60 Master ID Register (DMA_DCHMID60)	8	R/W	00h	on page 824
FC0A_017D	Channel 61 Master ID Register (DMA_DCHMID61)	8	R/W	00h	on page 824
FC0A_017E	Channel 62 Master ID Register (DMA_DCHMID62)	8	R/W	00h	on page 824
FC0A_017F	Channel 63 Master ID Register (DMA_DCHMID63)	8	R/W	00h	on page 824
FC0A_0180-FC0A_0FFF	Reserved				

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1000	TCD0 Source Address (DMA_TCD0_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1004	TCD0 Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	000Xh	on page 826
FC0A_1006	TCD0 Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	000Xh	on page 827
FC0A_1008	TCD0 Minor Byte Count (Minor Loop Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_100C	(DMA_TCD0_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1010	(DMA_TCD0_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1014	(DMA_TCD0_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD0 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1016	(DMA_TCD0_DOFF)	16	R/W	000Xh	on page 833
FC0A_1018	(DMA_TCD0_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_101C	(DMA_TCD0_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD0_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_101E	(DMA_TCD0_CSR)	16	R/W	000Xh	on page 836
FC0A_1020	TCD1 Source Address (DMA_TCD1_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1024	TCD1 Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	000Xh	on page 826
FC0A_1026	TCD1 Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	000Xh	on page 827
FC0A_1028	TCD1 Minor Byte Count (Minor Loop Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_102C	(DMA_TCD1_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1030	(DMA_TCD1_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1034	(DMA_TCD1_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD1 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_CITER_ELINKNO)	16	R/W	000Xh	on page 832

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1036	(DMA_TCD1_DOFF)	16	R/W	000Xh	on page 833
FC0A_1038	(DMA_TCD1_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_103C	(DMA_TCD1_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD1_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_103E	(DMA_TCD1_CSR)	16	R/W	000Xh	on page 836
FC0A_1040	TCD2 Source Address (DMA_TCD2_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1044	TCD2 Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	000Xh	on page 826
FC0A_1046	TCD2 Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	000Xh	on page 827
FC0A_1048	TCD2 Minor Byte Count (Minor Loop Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_104C	(DMA_TCD2_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1050	(DMA_TCD2_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1054	(DMA_TCD2_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD2 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1056	(DMA_TCD2_DOFF)	16	R/W	000Xh	on page 833
FC0A_1058	(DMA_TCD2_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_105C	(DMA_TCD2_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD2_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_105E	(DMA_TCD2_CSR)	16	R/W	000Xh	on page 836
FC0A_1060	TCD3 Source Address (DMA_TCD3_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1064	TCD3 Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	000Xh	on page 826
FC0A_1066	TCD3 Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	000Xh	on page 827
FC0A_1068	(DMA_TCD3_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_106C	(DMA_TCD3_SLAST)	32	R/W	0000_000Xh	on page 830

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1070	(DMA_TCD3_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1074	(DMA_TCD3_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD3 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1076	(DMA_TCD3_DOFF)	16	R/W	000Xh	on page 833
FC0A_1078	(DMA_TCD3_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_107C	(DMA_TCD3_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD3_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_107E	(DMA_TCD3_CSR)	16	R/W	000Xh	on page 836
FC0A_1080	TCD4 Source Address (DMA_TCD4_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1084	TCD4 Transfer Attributes (DMA_TCD4_ATTR)	16	R/W	000Xh	on page 826
FC0A_1086	TCD4 Signed Source Address Offset (DMA_TCD4_SOFF)	16	R/W	000Xh	on page 827
FC0A_1088	(DMA_TCD4_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD4_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD4_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_108C	(DMA_TCD4_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1090	(DMA_TCD4_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1094	(DMA_TCD4_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD4 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1096	(DMA_TCD4_DOFF)	16	R/W	000Xh	on page 833
FC0A_1098	(DMA_TCD4_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_109C	(DMA_TCD4_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD4_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_109E	(DMA_TCD4_CSR)	16	R/W	000Xh	on page 836
FC0A_10A0	TCD5 Source Address (DMA_TCD5_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_10A4	TCD5 Transfer Attributes (DMA_TCD5_ATTR)	16	R/W	000Xh	on page 826
FC0A_10A6	TCD5 Signed Source Address Offset (DMA_TCD5_SOFF)	16	R/W	000Xh	on page 827

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_10A8	(DMA_TCD5_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD5_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD5_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_10AC	(DMA_TCD5_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_10B0	(DMA_TCD5_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_10B4	(DMA_TCD5_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD5 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_10B6	(DMA_TCD5_DOFF)	16	R/W	000Xh	on page 833
FC0A_10B8	(DMA_TCD5_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_10BC	(DMA_TCD5_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD5_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_10BE	(DMA_TCD5_CSR)	16	R/W	000Xh	on page 836
FC0A_10C0	TCD6 Source Address (DMA_TCD6_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_10C4	TCD6 Transfer Attributes (DMA_TCD6_ATTR)	16	R/W	000Xh	on page 826
FC0A_10C6	TCD6 Signed Source Address Offset (DMA_TCD6_SOFF)	16	R/W	000Xh	on page 827
FC0A_10C8	(DMA_TCD6_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD6_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD6_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_10CC	(DMA_TCD6_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_10D0	(DMA_TCD6_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_10D4	(DMA_TCD6_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD6 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_10D6	(DMA_TCD6_DOFF)	16	R/W	000Xh	on page 833
FC0A_10D8	(DMA_TCD6_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_10DC	(DMA_TCD6_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD6_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_10DE	(DMA_TCD6_CSR)	16	R/W	000Xh	on page 836
FC0A_10E0	TCD7 Source Address (DMA_TCD7_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_10E4	TCD7 Transfer Attributes (DMA_TCD7_ATTR)	16	R/W	000Xh	on page 826
FC0A_10E6	TCD7 Signed Source Address Offset (DMA_TCD7_SOFF)	16	R/W	000Xh	on page 827
FC0A_10E8	(DMA_TCD7_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD7_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD7_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_10EC	(DMA_TCD7_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_10F0	(DMA_TCD7_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_10F4	(DMA_TCD7_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD7 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_10F6	(DMA_TCD7_DOFF)	16	R/W	000Xh	on page 833
FC0A_10F8	(DMA_TCD7_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_10FC	(DMA_TCD7_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD7_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_10FE	(DMA_TCD7_CSR)	16	R/W	000Xh	on page 836
FC0A_1100	TCD8 Source Address (DMA_TCD8_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1104	TCD8 Transfer Attributes (DMA_TCD8_ATTR)	16	R/W	000Xh	on page 826
FC0A_1106	TCD8 Signed Source Address Offset (DMA_TCD8_SOFF)	16	R/W	000Xh	on page 827
FC0A_1108	(DMA_TCD8_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD8_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD8_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_110C	(DMA_TCD8_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1110	(DMA_TCD8_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1114	(DMA_TCD8_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD8 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD8_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1116	(DMA_TCD8_DOFF)	16	R/W	000Xh	on page 833
FC0A_1118	(DMA_TCD8_DLASTSGA)	32	R/W	0000_000Xh	on page 834

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_111C	(DMA_TCD8_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD8_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_111E	(DMA_TCD8_CSR)	16	R/W	000Xh	on page 836
FC0A_1120	TCD9 Source Address (DMA_TCD9_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1124	TCD9 Transfer Attributes (DMA_TCD9_ATTR)	16	R/W	000Xh	on page 826
FC0A_1126	TCD9 Signed Source Address Offset (DMA_TCD9_SOFF)	16	R/W	000Xh	on page 827
FC0A_1128	(DMA_TCD9_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD9_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD9_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_112C	(DMA_TCD9_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1130	(DMA_TCD9_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1134	(DMA_TCD9_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD9 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD9_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1136	(DMA_TCD9_DOFF)	16	R/W	000Xh	on page 833
FC0A_1138	(DMA_TCD9_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_113C	(DMA_TCD9_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD9_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_113E	(DMA_TCD9_CSR)	16	R/W	000Xh	on page 836
FC0A_1140	TCD10 Source Address (DMA_TCD10_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1144	TCD10 Transfer Attributes (DMA_TCD10_ATTR)	16	R/W	000Xh	on page 826
FC0A_1146	TCD10 Signed Source Address Offset (DMA_TCD10_SOFF)	16	R/W	000Xh	on page 827
FC0A_1148	(DMA_TCD10_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD10_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD10_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_114C	(DMA_TCD10_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1150	(DMA_TCD10_DADDR)	32	R/W	0000_000Xh	on page 831

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1154	(DMA_TCD10_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD10 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD10_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1156	(DMA_TCD10_DOFF)	16	R/W	000Xh	on page 833
FC0A_1158	(DMA_TCD10_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_115C	(DMA_TCD10_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD10_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_115E	(DMA_TCD10_CSR)	16	R/W	000Xh	on page 836
FC0A_1160	TCD11 Source Address (DMA_TCD11_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1164	TCD11 Transfer Attributes (DMA_TCD11_ATTR)	16	R/W	000Xh	on page 826
FC0A_1166	TCD11 Signed Source Address Offset (DMA_TCD11_SOFF)	16	R/W	000Xh	on page 827
FC0A_1168	(DMA_TCD11_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD11_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD11_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_116C	(DMA_TCD11_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1170	(DMA_TCD11_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1174	(DMA_TCD11_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD11 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD11_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1176	(DMA_TCD11_DOFF)	16	R/W	000Xh	on page 833
FC0A_1178	(DMA_TCD11_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_117C	(DMA_TCD11_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD11_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_117E	(DMA_TCD11_CSR)	16	R/W	000Xh	on page 836
FC0A_1180	TCD12 Source Address (DMA_TCD12_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1184	TCD12 Transfer Attributes (DMA_TCD12_ATTR)	16	R/W	000Xh	on page 826
FC0A_1186	TCD12 Signed Source Address Offset (DMA_TCD12_SOFF)	16	R/W	000Xh	on page 827

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1188	(DMA_TCD12_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD12_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD12_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_118C	(DMA_TCD12_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1190	(DMA_TCD12_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1194	(DMA_TCD12_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD12 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD12_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1196	(DMA_TCD12_DOFF)	16	R/W	000Xh	on page 833
FC0A_1198	(DMA_TCD12_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_119C	(DMA_TCD12_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD12_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_119E	(DMA_TCD12_CSR)	16	R/W	000Xh	on page 836
FC0A_11A0	TCD13 Source Address (DMA_TCD13_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_11A4	TCD13 Transfer Attributes (DMA_TCD13_ATTR)	16	R/W	000Xh	on page 826
FC0A_11A6	TCD13 Signed Source Address Offset (DMA_TCD13_SOFF)	16	R/W	000Xh	on page 827
FC0A_11A8	(DMA_TCD13_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD13_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD13_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_11AC	(DMA_TCD13_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_11B0	(DMA_TCD13_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_11B4	(DMA_TCD13_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD13 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD13_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_11B6	(DMA_TCD13_DOFF)	16	R/W	000Xh	on page 833
FC0A_11B8	(DMA_TCD13_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_11BC	(DMA_TCD13_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD13_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_11BE	(DMA_TCD13_CSR)	16	R/W	000Xh	on page 836
FC0A_11C0	TCD14 Source Address (DMA_TCD14_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_11C4	TCD14 Transfer Attributes (DMA_TCD14_ATTR)	16	R/W	000Xh	on page 826
FC0A_11C6	TCD14 Signed Source Address Offset (DMA_TCD14_SOFF)	16	R/W	000Xh	on page 827
FC0A_11C8	(DMA_TCD14_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD14_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD14_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_11CC	(DMA_TCD14_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_11D0	(DMA_TCD14_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_11D4	(DMA_TCD14_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD14 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD14_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_11D6	(DMA_TCD14_DOFF)	16	R/W	000Xh	on page 833
FC0A_11D8	(DMA_TCD14_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_11DC	(DMA_TCD14_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD14_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_11DE	(DMA_TCD14_CSR)	16	R/W	000Xh	on page 836
FC0A_11E0	TCD15 Source Address (DMA_TCD15_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_11E4	TCD15 Transfer Attributes (DMA_TCD15_ATTR)	16	R/W	000Xh	on page 826
FC0A_11E6	TCD15 Signed Source Address Offset (DMA_TCD15_SOFF)	16	R/W	000Xh	on page 827
FC0A_11E8	(DMA_TCD15_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD15_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD15_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_11EC	(DMA_TCD15_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_11F0	(DMA_TCD15_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_11F4	(DMA_TCD15_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD15 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD15_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_11F6	(DMA_TCD15_DOFF)	16	R/W	000Xh	on page 833
FC0A_11F8	(DMA_TCD15_DLASTSGA)	32	R/W	0000_000Xh	on page 834

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_11FC	(DMA_TCD15_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD15_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_11FE	(DMA_TCD15_CSR)	16	R/W	000Xh	on page 836
FC0A_1200	TCD16 Source Address (DMA_TCD16_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1204	TCD16 Transfer Attributes (DMA_TCD16_ATTR)	16	R/W	000Xh	on page 826
FC0A_1206	TCD16 Signed Source Address Offset (DMA_TCD16_SOFF)	16	R/W	000Xh	on page 827
FC0A_1208	(DMA_TCD16_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD16_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD16_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_120C	(DMA_TCD16_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1210	(DMA_TCD16_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1214	(DMA_TCD16_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD16 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD16_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1216	(DMA_TCD16_DOFF)	16	R/W	000Xh	on page 833
FC0A_1218	(DMA_TCD16_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_121C	(DMA_TCD16_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD16_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_121E	(DMA_TCD16_CSR)	16	R/W	000Xh	on page 836
FC0A_1220	TCD17 Source Address (DMA_TCD17_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1224	TCD17 Transfer Attributes (DMA_TCD17_ATTR)	16	R/W	000Xh	on page 826
FC0A_1226	TCD17 Signed Source Address Offset (DMA_TCD17_SOFF)	16	R/W	000Xh	on page 827
FC0A_1228	(DMA_TCD17_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD17_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD17_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_122C	(DMA_TCD17_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1230	(DMA_TCD17_DADDR)	32	R/W	0000_000Xh	on page 831

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1234	(DMA_TCD17_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD17 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD17_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1236	(DMA_TCD17_DOFF)	16	R/W	000Xh	on page 833
FC0A_1238	(DMA_TCD17_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_123C	(DMA_TCD17_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD17_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_123E	(DMA_TCD17_CSR)	16	R/W	000Xh	on page 836
FC0A_1240	TCD18 Source Address (DMA_TCD18_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1244	TCD18 Transfer Attributes (DMA_TCD18_ATTR)	16	R/W	000Xh	on page 826
FC0A_1246	TCD18 Signed Source Address Offset (DMA_TCD18_SOFF)	16	R/W	000Xh	on page 827
FC0A_1248	(DMA_TCD18_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD18_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD18_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_124C	(DMA_TCD18_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1250	(DMA_TCD18_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1254	(DMA_TCD18_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD18 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD18_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1256	(DMA_TCD18_DOFF)	16	R/W	000Xh	on page 833
FC0A_1258	(DMA_TCD18_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_125C	(DMA_TCD18_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD18_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_125E	(DMA_TCD18_CSR)	16	R/W	000Xh	on page 836
FC0A_1260	TCD19 Source Address (DMA_TCD19_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1264	TCD19 Transfer Attributes (DMA_TCD19_ATTR)	16	R/W	000Xh	on page 826
FC0A_1266	TCD19 Signed Source Address Offset (DMA_TCD19_SOFF)	16	R/W	000Xh	on page 827

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1268	(DMA_TCD19_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD19_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD19_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_126C	(DMA_TCD19_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1270	(DMA_TCD19_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1274	(DMA_TCD19_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD19 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD19_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1276	(DMA_TCD19_DOFF)	16	R/W	000Xh	on page 833
FC0A_1278	(DMA_TCD19_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_127C	(DMA_TCD19_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD19_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_127E	(DMA_TCD19_CSR)	16	R/W	000Xh	on page 836
FC0A_1280	TCD20 Source Address (DMA_TCD20_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1284	TCD20 Transfer Attributes (DMA_TCD20_ATTR)	16	R/W	000Xh	on page 826
FC0A_1286	TCD20 Signed Source Address Offset (DMA_TCD20_SOFF)	16	R/W	000Xh	on page 827
FC0A_1288	(DMA_TCD20_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD20_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD20_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_128C	(DMA_TCD20_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1290	(DMA_TCD20_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1294	(DMA_TCD20_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD20 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD20_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1296	(DMA_TCD20_DOFF)	16	R/W	000Xh	on page 833
FC0A_1298	(DMA_TCD20_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_129C	(DMA_TCD20_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD20_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_129E	(DMA_TCD20_CSR)	16	R/W	000Xh	on page 836
FC0A_12A0	TCD21 Source Address (DMA_TCD21_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_12A4	TCD21 Transfer Attributes (DMA_TCD21_ATTR)	16	R/W	000Xh	on page 826
FC0A_12A6	TCD21 Signed Source Address Offset (DMA_TCD21_SOFF)	16	R/W	000Xh	on page 827
FC0A_12A8	(DMA_TCD21_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD21_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD21_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_12AC	(DMA_TCD21_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_12B0	(DMA_TCD21_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_12B4	(DMA_TCD21_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD21 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD21_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_12B6	(DMA_TCD21_DOFF)	16	R/W	000Xh	on page 833
FC0A_12B8	(DMA_TCD21_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_12BC	(DMA_TCD21_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD21_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_12BE	(DMA_TCD21_CSR)	16	R/W	000Xh	on page 836
FC0A_12C0	TCD22 Source Address (DMA_TCD22_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_12C4	TCD22 Transfer Attributes (DMA_TCD22_ATTR)	16	R/W	000Xh	on page 826
FC0A_12C6	TCD22 Signed Source Address Offset (DMA_TCD22_SOFF)	16	R/W	000Xh	on page 827
FC0A_12C8	(DMA_TCD22_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD22_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD22_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_12CC	(DMA_TCD22_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_12D0	(DMA_TCD22_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_12D4	(DMA_TCD22_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD22 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD22_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_12D6	(DMA_TCD22_DOFF)	16	R/W	000Xh	on page 833
FC0A_12D8	(DMA_TCD22_DLASTSGA)	32	R/W	0000_000Xh	on page 834

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_12DC	(DMA_TCD22_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD22_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_12DE	(DMA_TCD22_CSR)	16	R/W	000Xh	on page 836
FC0A_12E0	TCD23 Source Address (DMA_TCD23_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_12E4	TCD23 Transfer Attributes (DMA_TCD23_ATTR)	16	R/W	000Xh	on page 826
FC0A_12E6	TCD23 Signed Source Address Offset (DMA_TCD23_SOFF)	16	R/W	000Xh	on page 827
FC0A_12E8	(DMA_TCD23_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD23_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD23_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_12EC	(DMA_TCD23_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_12F0	(DMA_TCD23_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_12F4	(DMA_TCD23_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD23 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD23_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_12F6	(DMA_TCD23_DOFF)	16	R/W	000Xh	on page 833
FC0A_12F8	(DMA_TCD23_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_12FC	(DMA_TCD23_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD23_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_12FE	(DMA_TCD23_CSR)	16	R/W	000Xh	on page 836
FC0A_1300	TCD24 Source Address (DMA_TCD24_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1304	TCD24 Transfer Attributes (DMA_TCD24_ATTR)	16	R/W	000Xh	on page 826
FC0A_1306	TCD24 Signed Source Address Offset (DMA_TCD24_SOFF)	16	R/W	000Xh	on page 827
FC0A_1308	(DMA_TCD24_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD24_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD24_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_130C	(DMA_TCD24_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1310	(DMA_TCD24_DADDR)	32	R/W	0000_000Xh	on page 831

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1314	(DMA_TCD24_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD24 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD24_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1316	(DMA_TCD24_DOFF)	16	R/W	000Xh	on page 833
FC0A_1318	(DMA_TCD24_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_131C	(DMA_TCD24_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD24_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_131E	(DMA_TCD24_CSR)	16	R/W	000Xh	on page 836
FC0A_1320	TCD25 Source Address (DMA_TCD25_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1324	TCD25 Transfer Attributes (DMA_TCD25_ATTR)	16	R/W	000Xh	on page 826
FC0A_1326	TCD25 Signed Source Address Offset (DMA_TCD25_SOFF)	16	R/W	000Xh	on page 827
FC0A_1328	(DMA_TCD25_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD25_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD25_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_132C	(DMA_TCD25_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1330	(DMA_TCD25_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1334	(DMA_TCD25_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD25 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD25_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1336	(DMA_TCD25_DOFF)	16	R/W	000Xh	on page 833
FC0A_1338	(DMA_TCD25_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_133C	(DMA_TCD25_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD25_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_133E	(DMA_TCD25_CSR)	16	R/W	000Xh	on page 836
FC0A_1340	TCD26 Source Address (DMA_TCD26_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1344	TCD26 Transfer Attributes (DMA_TCD26_ATTR)	16	R/W	000Xh	on page 826
FC0A_1346	TCD26 Signed Source Address Offset (DMA_TCD26_SOFF)	16	R/W	000Xh	on page 827

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1348	(DMA_TCD26_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD26_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD26_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_134C	(DMA_TCD26_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1350	(DMA_TCD26_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1354	(DMA_TCD26_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD26 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD26_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1356	(DMA_TCD26_DOFF)	16	R/W	000Xh	on page 833
FC0A_1358	(DMA_TCD26_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_135C	(DMA_TCD26_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD26_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_135E	(DMA_TCD26_CSR)	16	R/W	000Xh	on page 836
FC0A_1360	TCD27 Source Address (DMA_TCD27_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1364	TCD27 Transfer Attributes (DMA_TCD27_ATTR)	16	R/W	000Xh	on page 826
FC0A_1366	TCD27 Signed Source Address Offset (DMA_TCD27_SOFF)	16	R/W	000Xh	on page 827
FC0A_1368	(DMA_TCD27_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD27_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD27_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_136C	(DMA_TCD27_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1370	(DMA_TCD27_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1374	(DMA_TCD27_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD27 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD27_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1376	(DMA_TCD27_DOFF)	16	R/W	000Xh	on page 833
FC0A_1378	(DMA_TCD27_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_137C	(DMA_TCD27_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD27_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_137E	(DMA_TCD27_CSR)	16	R/W	000Xh	on page 836
FC0A_1380	TCD28 Source Address (DMA_TCD28_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1384	TCD28 Transfer Attributes (DMA_TCD28_ATTR)	16	R/W	000Xh	on page 826
FC0A_1386	TCD28 Signed Source Address Offset (DMA_TCD28_SOFF)	16	R/W	000Xh	on page 827
FC0A_1388	(DMA_TCD28_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD28_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD28_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_138C	(DMA_TCD28_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1390	(DMA_TCD28_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1394	(DMA_TCD28_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD28 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD28_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1396	(DMA_TCD28_DOFF)	16	R/W	000Xh	on page 833
FC0A_1398	(DMA_TCD28_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_139C	(DMA_TCD28_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD28_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_139E	(DMA_TCD28_CSR)	16	R/W	000Xh	on page 836
FC0A_13A0	TCD29 Source Address (DMA_TCD29_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_13A4	TCD29 Transfer Attributes (DMA_TCD29_ATTR)	16	R/W	000Xh	on page 826
FC0A_13A6	TCD29 Signed Source Address Offset (DMA_TCD29_SOFF)	16	R/W	000Xh	on page 827
FC0A_13A8	(DMA_TCD29_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD29_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD29_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_13AC	(DMA_TCD29_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_13B0	(DMA_TCD29_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_13B4	(DMA_TCD29_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD29 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD29_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_13B6	(DMA_TCD29_DOFF)	16	R/W	000Xh	on page 833
FC0A_13B8	(DMA_TCD29_DLASTSGA)	32	R/W	0000_000Xh	on page 834

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_13BC	(DMA_TCD29_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD29_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_13BE	(DMA_TCD29_CSR)	16	R/W	000Xh	on page 836
FC0A_13C0	TCD30 Source Address (DMA_TCD30_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_13C4	TCD30 Transfer Attributes (DMA_TCD30_ATTR)	16	R/W	000Xh	on page 826
FC0A_13C6	TCD30 Signed Source Address Offset (DMA_TCD30_SOFF)	16	R/W	000Xh	on page 827
FC0A_13C8	(DMA_TCD30_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD30_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD30_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_13CC	(DMA_TCD30_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_13D0	(DMA_TCD30_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_13D4	(DMA_TCD30_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD30 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD30_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_13D6	(DMA_TCD30_DOFF)	16	R/W	000Xh	on page 833
FC0A_13D8	(DMA_TCD30_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_13DC	(DMA_TCD30_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD30_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_13DE	(DMA_TCD30_CSR)	16	R/W	000Xh	on page 836
FC0A_13E0	TCD31 Source Address (DMA_TCD31_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_13E4	TCD31 Transfer Attributes (DMA_TCD31_ATTR)	16	R/W	000Xh	on page 826
FC0A_13E6	TCD31 Signed Source Address Offset (DMA_TCD31_SOFF)	16	R/W	000Xh	on page 827
FC0A_13E8	(DMA_TCD31_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD31_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD31_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_13EC	(DMA_TCD31_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_13F0	(DMA_TCD31_DADDR)	32	R/W	0000_000Xh	on page 831

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_13F4	(DMA_TCD31_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD31 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD31_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_13F6	(DMA_TCD31_DOFF)	16	R/W	000Xh	on page 833
FC0A_13F8	(DMA_TCD31_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_13FC	(DMA_TCD31_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD31_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_13FE	(DMA_TCD31_CSR)	16	R/W	000Xh	on page 836
FC0A_1400	TCD32 Source Address (DMA_TCD32_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1404	TCD32 Transfer Attributes (DMA_TCD32_ATTR)	16	R/W	000Xh	on page 826
FC0A_1406	TCD32 Signed Source Address Offset (DMA_TCD32_SOFF)	16	R/W	000Xh	on page 827
FC0A_1408	(DMA_TCD32_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD32_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD32_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_140C	(DMA_TCD32_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1410	(DMA_TCD32_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1414	(DMA_TCD32_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD32 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD32_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1416	(DMA_TCD32_DOFF)	16	R/W	000Xh	on page 833
FC0A_1418	(DMA_TCD32_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_141C	(DMA_TCD32_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD32_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_141E	(DMA_TCD32_CSR)	16	R/W	000Xh	on page 836
FC0A_1420	TCD33 Source Address (DMA_TCD33_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1424	TCD33 Transfer Attributes (DMA_TCD33_ATTR)	16	R/W	000Xh	on page 826
FC0A_1426	TCD33 Signed Source Address Offset (DMA_TCD33_SOFF)	16	R/W	000Xh	on page 827

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1428	(DMA_TCD33_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD33_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD33_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_142C	(DMA_TCD33_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1430	(DMA_TCD33_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1434	(DMA_TCD33_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD33 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD33_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1436	(DMA_TCD33_DOFF)	16	R/W	000Xh	on page 833
FC0A_1438	(DMA_TCD33_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_143C	(DMA_TCD33_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD33_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_143E	(DMA_TCD33_CSR)	16	R/W	000Xh	on page 836
FC0A_1440	TCD34 Source Address (DMA_TCD34_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1444	TCD34 Transfer Attributes (DMA_TCD34_ATTR)	16	R/W	000Xh	on page 826
FC0A_1446	TCD34 Signed Source Address Offset (DMA_TCD34_SOFF)	16	R/W	000Xh	on page 827
FC0A_1448	(DMA_TCD34_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD34_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD34_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_144C	(DMA_TCD34_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1450	(DMA_TCD34_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1454	(DMA_TCD34_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD34 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD34_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1456	(DMA_TCD34_DOFF)	16	R/W	000Xh	on page 833
FC0A_1458	(DMA_TCD34_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_145C	(DMA_TCD34_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD34_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_145E	(DMA_TCD34_CSR)	16	R/W	000Xh	on page 836
FC0A_1460	TCD35 Source Address (DMA_TCD35_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1464	TCD35 Transfer Attributes (DMA_TCD35_ATTR)	16	R/W	000Xh	on page 826
FC0A_1466	TCD35 Signed Source Address Offset (DMA_TCD35_SOFF)	16	R/W	000Xh	on page 827
FC0A_1468	(DMA_TCD35_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD35_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD35_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_146C	(DMA_TCD35_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1470	(DMA_TCD35_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1474	(DMA_TCD35_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD35 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD35_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1476	(DMA_TCD35_DOFF)	16	R/W	000Xh	on page 833
FC0A_1478	(DMA_TCD35_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_147C	(DMA_TCD35_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD35_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_147E	(DMA_TCD35_CSR)	16	R/W	000Xh	on page 836
FC0A_1480	TCD36 Source Address (DMA_TCD36_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1484	TCD36 Transfer Attributes (DMA_TCD36_ATTR)	16	R/W	000Xh	on page 826
FC0A_1486	TCD36 Signed Source Address Offset (DMA_TCD36_SOFF)	16	R/W	000Xh	on page 827
FC0A_1488	(DMA_TCD36_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD36_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD36_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_148C	(DMA_TCD36_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1490	(DMA_TCD36_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1494	(DMA_TCD36_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD36 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD36_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1496	(DMA_TCD36_DOFF)	16	R/W	000Xh	on page 833
FC0A_1498	(DMA_TCD36_DLASTSGA)	32	R/W	0000_000Xh	on page 834

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_149C	(DMA_TCD36_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD36_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_149E	(DMA_TCD36_CSR)	16	R/W	000Xh	on page 836
FC0A_14A0	TCD37 Source Address (DMA_TCD37_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_14A4	TCD37 Transfer Attributes (DMA_TCD37_ATTR)	16	R/W	000Xh	on page 826
FC0A_14A6	TCD37 Signed Source Address Offset (DMA_TCD37_SOFF)	16	R/W	000Xh	on page 827
FC0A_14A8	(DMA_TCD37_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD37_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD37_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_14AC	(DMA_TCD37_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_14B0	(DMA_TCD37_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_14B4	(DMA_TCD37_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD37 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD37_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_14B6	(DMA_TCD37_DOFF)	16	R/W	000Xh	on page 833
FC0A_14B8	(DMA_TCD37_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_14BC	(DMA_TCD37_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD37_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_14BE	(DMA_TCD37_CSR)	16	R/W	000Xh	on page 836
FC0A_14C0	TCD38 Source Address (DMA_TCD38_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_14C4	TCD38 Transfer Attributes (DMA_TCD38_ATTR)	16	R/W	000Xh	on page 826
FC0A_14C6	TCD38 Signed Source Address Offset (DMA_TCD38_SOFF)	16	R/W	000Xh	on page 827
FC0A_14C8	(DMA_TCD38_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD38_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD38_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_14CC	(DMA_TCD38_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_14D0	(DMA_TCD38_DADDR)	32	R/W	0000_000Xh	on page 831

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_14D4	(DMA_TCD38_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD38 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD38_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_14D6	(DMA_TCD38_DOFF)	16	R/W	000Xh	on page 833
FC0A_14D8	(DMA_TCD38_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_14DC	(DMA_TCD38_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD38_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_14DE	(DMA_TCD38_CSR)	16	R/W	000Xh	on page 836
FC0A_14E0	TCD39 Source Address (DMA_TCD39_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_14E4	TCD39 Transfer Attributes (DMA_TCD39_ATTR)	16	R/W	000Xh	on page 826
FC0A_14E6	TCD39 Signed Source Address Offset (DMA_TCD39_SOFF)	16	R/W	000Xh	on page 827
FC0A_14E8	(DMA_TCD39_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD39_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD39_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_14EC	(DMA_TCD39_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_14F0	(DMA_TCD39_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_14F4	(DMA_TCD39_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD39 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD39_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_14F6	(DMA_TCD39_DOFF)	16	R/W	000Xh	on page 833
FC0A_14F8	(DMA_TCD39_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_14FC	(DMA_TCD39_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD39_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_14FE	(DMA_TCD39_CSR)	16	R/W	000Xh	on page 836
FC0A_1500	TCD40 Source Address (DMA_TCD40_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1504	TCD40 Transfer Attributes (DMA_TCD40_ATTR)	16	R/W	000Xh	on page 826
FC0A_1506	TCD40 Signed Source Address Offset (DMA_TCD40_SOFF)	16	R/W	000Xh	on page 827

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1508	(DMA_TCD40_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD40_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD40_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_150C	(DMA_TCD40_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1510	(DMA_TCD40_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1514	(DMA_TCD40_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD40 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD40_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1516	(DMA_TCD40_DOFF)	16	R/W	000Xh	on page 833
FC0A_1518	(DMA_TCD40_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_151C	(DMA_TCD40_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD40_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_151E	(DMA_TCD40_CSR)	16	R/W	000Xh	on page 836
FC0A_1520	TCD41 Source Address (DMA_TCD41_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1524	TCD41 Transfer Attributes (DMA_TCD41_ATTR)	16	R/W	000Xh	on page 826
FC0A_1526	TCD41 Signed Source Address Offset (DMA_TCD41_SOFF)	16	R/W	000Xh	on page 827
FC0A_1528	(DMA_TCD41_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD41_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD41_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_152C	(DMA_TCD41_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1530	(DMA_TCD41_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1534	(DMA_TCD41_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD41 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) DMA_TCD41_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1536	(DMA_TCD41_DOFF)	16	R/W	000Xh	on page 833
FC0A_1538	(DMA_TCD41_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_153C	(DMA_TCD41_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD41_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_153E	(DMA_TCD41_CSR)	16	R/W	000Xh	on page 836
FC0A_1540	TCD42 Source Address (DMA_TCD42_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1544	TCD42 Transfer Attributes (DMA_TCD42_ATTR)	16	R/W	000Xh	on page 826
FC0A_1546	TCD42 Signed Source Address Offset (DMA_TCD42_SOFF)	16	R/W	000Xh	on page 827
FC0A_1548	(DMA_TCD42_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD42_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD42_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_154C	(DMA_TCD42_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1550	(DMA_TCD42_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1554	(DMA_TCD42_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD42 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD42_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1556	(DMA_TCD42_DOFF)	16	R/W	000Xh	on page 833
FC0A_1558	(DMA_TCD42_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_155C	(DMA_TCD42_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD42_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_155E	(DMA_TCD42_CSR)	16	R/W	000Xh	on page 836
FC0A_1560	TCD43 Source Address (DMA_TCD43_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1564	TCD43 Transfer Attributes (DMA_TCD43_ATTR)	16	R/W	000Xh	on page 826
FC0A_1566	TCD43 Signed Source Address Offset (DMA_TCD43_SOFF)	16	R/W	000Xh	on page 827
FC0A_1568	(DMA_TCD43_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD43_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD43_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_156C	(DMA_TCD43_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1570	(DMA_TCD43_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1574	(DMA_TCD43_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD43 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD43_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1576	(DMA_TCD43_DOFF)	16	R/W	000Xh	on page 833
FC0A_1578	(DMA_TCD43_DLASTSGA)	32	R/W	0000_000Xh	on page 834

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_157C	(DMA_TCD43_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD43_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_157E	(DMA_TCD43_CSR)	16	R/W	000Xh	on page 836
FC0A_1580	TCD44 Source Address (DMA_TCD44_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1584	TCD44 Transfer Attributes (DMA_TCD44_ATTR)	16	R/W	000Xh	on page 826
FC0A_1586	TCD44 Signed Source Address Offset (DMA_TCD44_SOFF)	16	R/W	000Xh	on page 827
FC0A_1588	(DMA_TCD44_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD44_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD44_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_158C	(DMA_TCD44_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1590	(DMA_TCD44_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1594	(DMA_TCD44_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD44 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD44_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1596	(DMA_TCD44_DOFF)	16	R/W	000Xh	on page 833
FC0A_1598	(DMA_TCD44_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_159C	(DMA_TCD44_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD44_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_159E	(DMA_TCD44_CSR)	16	R/W	000Xh	on page 836
FC0A_15A0	TCD45 Source Address (DMA_TCD45_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_15A4	TCD45 Transfer Attributes (DMA_TCD45_ATTR)	16	R/W	000Xh	on page 826
FC0A_15A6	TCD45 Signed Source Address Offset (DMA_TCD45_SOFF)	16	R/W	000Xh	on page 827
FC0A_15A8	(DMA_TCD45_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD45_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD45_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_15AC	(DMA_TCD45_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_15B0	(DMA_TCD45_DADDR)	32	R/W	0000_000Xh	on page 831

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_15B4	(DMA_TCD45_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD45 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) DMA_TCD45_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_15B6	(DMA_TCD45_DOFF)	16	R/W	000Xh	on page 833
FC0A_15B8	(DMA_TCD45_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_15BC	(DMA_TCD45_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD45_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_15BE	(DMA_TCD45_CSR)	16	R/W	000Xh	on page 836
FC0A_15C0	TCD46 Source Address (DMA_TCD46_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_15C4	TCD46 Transfer Attributes (DMA_TCD46_ATTR)	16	R/W	000Xh	on page 826
FC0A_15C6	TCD46 Signed Source Address Offset (DMA_TCD46_SOFF)	16	R/W	000Xh	on page 827
FC0A_15C8	(DMA_TCD46_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD46_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD46_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_15CC	(DMA_TCD46_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_15D0	(DMA_TCD46_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_15D4	(DMA_TCD46_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD46 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) DMA_TCD46_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_15D6	(DMA_TCD46_DOFF)	16	R/W	000Xh	on page 833
FC0A_15D8	(DMA_TCD46_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_15DC	(DMA_TCD46_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD46_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_15DE	(DMA_TCD46_CSR)	16	R/W	000Xh	on page 836
FC0A_15E0	TCD47 Source Address (DMA_TCD47_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_15E4	TCD47 Transfer Attributes (DMA_TCD47_ATTR)	16	R/W	000Xh	on page 826
FC0A_15E6	TCD47 Signed Source Address Offset (DMA_TCD47_SOFF)	16	R/W	000Xh	on page 826

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_15E8	(DMA_TCD47_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD47_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD47_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 828
FC0A_15EC	(DMA_TCD47_SLAST)	32	R/W	0000_000Xh	on page 829
FC0A_15F0	(DMA_TCD47_DADDR)	32	R/W	0000_000Xh	on page 830
FC0A_15F4	(DMA_TCD47_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD47 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) DMA_TCD47_CITER_ELINKNO	16	R/W	000Xh	on page 831
FC0A_15F6	(DMA_TCD47_DOFF)	16	R/W	000Xh	on page 832
FC0A_15F8	(DMA_TCD47_DLASTSGA)	32	R/W	0000_000Xh	on page 833
FC0A_15FC	(DMA_TCD47_BITER_ELINKYES)	16	R/W	000Xh	on page 834
	(DMA_TCD47_BITER_ELINKNO)	16	R/W	000Xh	on page 835
FC0A_15FE	(DMA_TCD47_CSR)	16	R/W	000Xh	on page 836
FC0A_1600	TCD48 Source Address (DMA_TCD48_SADDR)	32	R/W	0000_000Xh	on page 836
FC0A_1604	TCD48 Transfer Attributes (DMA_TCD48_ATTR)	16	R/W	000Xh	on page 825
FC0A_1606	TCD48 Signed Source Address Offset (DMA_TCD48_SOFF)	16	R/W	000Xh	on page 827
FC0A_1608	(DMA_TCD48_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD48_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD48_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_160C	(DMA_TCD48_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1610	(DMA_TCD48_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1614	(DMA_TCD48_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD48 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) DMA_TCD48_CITER_ELINKNO	16	R/W	000Xh	on page 832
FC0A_1616	(DMA_TCD48_DOFF)	16	R/W	000Xh	on page 833
FC0A_1618	(DMA_TCD48_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_161C	(DMA_TCD48_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD48_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_161E	(DMA_TCD48_CSR)	16	R/W	000Xh	on page 836
FC0A_1620	TCD49 Source Address (DMA_TCD49_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1624	TCD49 Transfer Attributes (DMA_TCD49_ATTR)	16	R/W	000Xh	on page 826
FC0A_1626	TCD49 Signed Source Address Offset (DMA_TCD49_SOFF)	16	R/W	000Xh	on page 827
FC0A_1628	(DMA_TCD49_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD49_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD49_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_162C	(DMA_TCD49_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1630	(DMA_TCD49_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1634	(DMA_TCD49_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD49 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD49_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1636	(DMA_TCD49_DOFF)	16	R/W	000Xh	on page 833
FC0A_1638	(DMA_TCD49_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_163C	(DMA_TCD49_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD49_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_163E	(DMA_TCD49_CSR)	16	R/W	000Xh	on page 836
FC0A_1640	TCD50 Source Address (DMA_TCD50_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1644	TCD50 Transfer Attributes (DMA_TCD50_ATTR)	16	R/W	000Xh	on page 826
FC0A_1646	TCD50 Signed Source Address Offset (DMA_TCD50_SOFF)	16	R/W	000Xh	on page 827
FC0A_1648	(DMA_TCD50_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD50_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD50_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_164C	(DMA_TCD50_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1650	(DMA_TCD50_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1654	(DMA_TCD50_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD50 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD50_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1656	(DMA_TCD50_DOFF)	16	R/W	000Xh	on page 833
FC0A_1658	(DMA_TCD50_DLASTSGA)	32	R/W	0000_000Xh	on page 834

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_165C	(DMA_TCD50_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD50_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_165E	(DMA_TCD50_CSR)	16	R/W	000Xh	on page 836
FC0A_1660	TCD51 Source Address (DMA_TCD51_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1664	TCD51 Transfer Attributes (DMA_TCD51_ATTR)	16	R/W	000Xh	on page 826
FC0A_1666	TCD51 Signed Source Address Offset (DMA_TCD51_SOFF)	16	R/W	000Xh	on page 827
FC0A_1668	(DMA_TCD51_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD51_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD51_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_166C	(DMA_TCD51_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1670	(DMA_TCD51_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1674	(DMA_TCD51_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD51 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD51_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1676	(DMA_TCD51_DOFF)	16	R/W	000Xh	on page 833
FC0A_1678	(DMA_TCD51_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_167C	(DMA_TCD51_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD51_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_167E	(DMA_TCD51_CSR)	16	R/W	000Xh	on page 836
FC0A_1680	TCD52 Source Address (DMA_TCD52_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1684	TCD52 Transfer Attributes (DMA_TCD52_ATTR)	16	R/W	000Xh	on page 826
FC0A_1686	TCD52 Signed Source Address Offset (DMA_TCD52_SOFF)	16	R/W	000Xh	on page 827
FC0A_1688	(DMA_TCD52_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD52_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD52_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_168C	(DMA_TCD52_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1690	(DMA_TCD52_DADDR)	32	R/W	0000_000Xh	on page 831

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1694	(DMA_TCD52_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD52 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD52_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1696	(DMA_TCD52_DOFF)	16	R/W	000Xh	on page 833
FC0A_1698	(DMA_TCD52_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_169C	(DMA_TCD52_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD52_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_169E	(DMA_TCD52_CSR)	16	R/W	000Xh	on page 836
FC0A_16A0	TCD53 Source Address (DMA_TCD53_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_16A4	TCD53 Transfer Attributes (DMA_TCD53_ATTR)	16	R/W	000Xh	on page 826
FC0A_16A6	TCD53 Signed Source Address Offset (DMA_TCD53_SOFF)	16	R/W	000Xh	on page 827
FC0A_16A8	(DMA_TCD53_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD53_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD53_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_16AC	(DMA_TCD53_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_16B0	(DMA_TCD53_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_16B4	(DMA_TCD53_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD53 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD53_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_16B6	(DMA_TCD53_DOFF)	16	R/W	000Xh	on page 833
FC0A_16B8	(DMA_TCD53_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_16BC	(DMA_TCD53_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD53_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_16BE	(DMA_TCD53_CSR)	16	R/W	000Xh	on page 836
FC0A_16C0	TCD54 Source Address (DMA_TCD54_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_16C4	TCD54 Transfer Attributes (DMA_TCD54_ATTR)	16	R/W	000Xh	on page 826
FC0A_16C6	TCD54 Signed Source Address Offset (DMA_TCD54_SOFF)	16	R/W	000Xh	on page 827

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_16C8	(DMA_TCD54_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD54_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD54_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_16CC	(DMA_TCD54_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_16D0	(DMA_TCD54_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_16D4	(DMA_TCD54_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD54 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD54_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_16D6	(DMA_TCD54_DOFF)	16	R/W	000Xh	on page 833
FC0A_16D8	(DMA_TCD54_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_16DC	(DMA_TCD54_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD54_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_16DE	(DMA_TCD54_CSR)	16	R/W	000Xh	on page 836
FC0A_16E0	TCD55 Source Address (DMA_TCD55_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_16E4	TCD55 Transfer Attributes (DMA_TCD55_ATTR)	16	R/W	000Xh	on page 826
FC0A_16E6	TCD55 Signed Source Address Offset (DMA_TCD55_SOFF)	16	R/W	000Xh	on page 827
FC0A_16E8	(DMA_TCD55_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD55_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD55_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_16EC	(DMA_TCD55_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_16F0	(DMA_TCD55_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_16F4	(DMA_TCD55_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD55 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD55_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_16F6	(DMA_TCD55_DOFF)	16	R/W	000Xh	on page 833
FC0A_16F8	(DMA_TCD55_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_16FC	(DMA_TCD55_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD55_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_16FE	(DMA_TCD55_CSR)	16	R/W	000Xh	on page 836
FC0A_1700	TCD56 Source Address (DMA_TCD56_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1704	TCD56 Transfer Attributes (DMA_TCD56_ATTR)	16	R/W	000Xh	on page 826
FC0A_1706	TCD56 Signed Source Address Offset (DMA_TCD56_SOFF)	16	R/W	000Xh	on page 827
FC0A_1708	(DMA_TCD56_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD56_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD56_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_170C	(DMA_TCD56_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1710	(DMA_TCD56_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1714	(DMA_TCD56_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD56 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD56_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1716	(DMA_TCD56_DOFF)	16	R/W	000Xh	on page 833
FC0A_1718	(DMA_TCD56_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_171C	(DMA_TCD56_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD56_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_171E	(DMA_TCD56_CSR)	16	R/W	000Xh	on page 836
FC0A_1720	TCD57 Source Address (DMA_TCD57_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1724	TCD57 Transfer Attributes (DMA_TCD57_ATTR)	16	R/W	000Xh	on page 826
FC0A_1726	TCD57 Signed Source Address Offset (DMA_TCD57_SOFF)	16	R/W	000Xh	on page 827
FC0A_1728	(DMA_TCD57_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD57_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD57_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_172C	(DMA_TCD57_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1730	(DMA_TCD57_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1734	(DMA_TCD57_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD57 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD57_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1736	(DMA_TCD57_DOFF)	16	R/W	000Xh	on page 833
FC0A_1738	(DMA_TCD57_DLASTSGA)	32	R/W	0000_000Xh	on page 834

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_173C	(DMA_TCD57_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD57_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_173E	(DMA_TCD57_CSR)	16	R/W	000Xh	on page 836
FC0A_1740	TCD58 Source Address (DMA_TCD58_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1744	TCD58 Transfer Attributes (DMA_TCD58_ATTR)	16	R/W	000Xh	on page 826
FC0A_1746	TCD58 Signed Source Address Offset (DMA_TCD58_SOFF)	16	R/W	000Xh	on page 827
FC0A_1748	(DMA_TCD58_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD58_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD58_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_174C	(DMA_TCD58_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1750	(DMA_TCD58_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1754	(DMA_TCD58_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD58 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD58_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1756	(DMA_TCD58_DOFF)	16	R/W	000Xh	on page 833
FC0A_1758	(DMA_TCD58_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_175C	(DMA_TCD58_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD58_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_175E	(DMA_TCD58_CSR)	16	R/W	000Xh	on page 836
FC0A_1760	TCD59 Source Address (DMA_TCD59_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1764	TCD59 Transfer Attributes (DMA_TCD59_ATTR)	16	R/W	000Xh	on page 826
FC0A_1766	TCD59 Signed Source Address Offset (DMA_TCD59_SOFF)	16	R/W	000Xh	on page 827
FC0A_1768	(DMA_TCD59_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD59_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD59_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_176C	(DMA_TCD59_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1770	(DMA_TCD59_DADDR)	32	R/W	0000_000Xh	on page 831

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1774	(DMA_TCD59_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD59 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD59_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1776	(DMA_TCD59_DOFF)	16	R/W	000Xh	on page 833
FC0A_1778	(DMA_TCD59_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_177C	(DMA_TCD59_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD59_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_177E	(DMA_TCD59_CSR)	16	R/W	000Xh	on page 836
FC0A_1780	TCD60 Source Address (DMA_TCD60_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_1784	TCD60 Transfer Attributes (DMA_TCD60_ATTR)	16	R/W	000Xh	on page 826
FC0A_1786	TCD60 Signed Source Address Offset (DMA_TCD60_SOFF)	16	R/W	000Xh	on page 827
FC0A_1788	(DMA_TCD60_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD60_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD60_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_178C	(DMA_TCD60_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_1790	(DMA_TCD60_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_1794	(DMA_TCD60_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD60 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD60_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_1796	(DMA_TCD60_DOFF)	16	R/W	000Xh	on page 833
FC0A_1798	(DMA_TCD60_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_179C	(DMA_TCD60_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD60_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_179E	(DMA_TCD60_CSR)	16	R/W	000Xh	on page 836
FC0A_17A0	TCD61 Source Address (DMA_TCD61_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_17A4	TCD61 Transfer Attributes (DMA_TCD61_ATTR)	16	R/W	000Xh	on page 826
FC0A_17A6	TCD61 Signed Source Address Offset (DMA_TCD61_SOFF)	16	R/W	000Xh	on page 827

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_17A8	(DMA_TCD61_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD61_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD61_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_17AC	(DMA_TCD61_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_17B0	(DMA_TCD61_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_17B4	(DMA_TCD61_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD61 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD61_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_17B6	(DMA_TCD61_DOFF)	16	R/W	000Xh	on page 833
FC0A_17B8	(DMA_TCD61_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_17BC	(DMA_TCD61_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD61_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_17BE	(DMA_TCD61_CSR)	16	R/W	000Xh	on page 836
FC0A_17C0	TCD62 Source Address (DMA_TCD62_SADDR)	32	R/W	0000_000Xh	on page 825
FC0A_17C4	TCD62 Transfer Attributes (DMA_TCD62_ATTR)	16	R/W	000Xh	on page 826
FC0A_17C6	TCD62 Signed Source Address Offset (DMA_TCD62_SOFF)	16	R/W	000Xh	on page 827
FC0A_17C8	(DMA_TCD62_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD62_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD62_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_17CC	(DMA_TCD62_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_17D0	(DMA_TCD62_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_17D4	(DMA_TCD62_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD62 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD62_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_17D6	(DMA_TCD62_DOFF)	16	R/W	000Xh	on page 833
FC0A_17D8	(DMA_TCD62_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_17DC	(DMA_TCD62_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD62_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_17DE	(DMA_TCD62_CSR)	16	R/W	000Xh	on page 836
FC0A_17E0	TCD63 Source Address (DMA_TCD63_SADDR)	32	R/W	0000_000Xh	on page 825

Table 290. DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_17E4	TCD63 Transfer Attributes (DMA_TCD63_ATTR)	16	R/W	000Xh	on page 826
FC0A_17E6	TCD63 Signed Source Address Offset (DMA_TCD63_SOFF)	16	R/W	000Xh	on page 826
FC0A_17E8	(DMA_TCD63_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 827
	(DMA_TCD63_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 828
	(DMA_TCD63_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 829
FC0A_17EC	(DMA_TCD63_SLAST)	32	R/W	0000_000Xh	on page 830
FC0A_17F0	(DMA_TCD63_DADDR)	32	R/W	0000_000Xh	on page 831
FC0A_17F4	(DMA_TCD63_CITER_ELINKYES)	16	R/W	000Xh	on page 831
	TCD63 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD63_CITER_ELINKNO)	16	R/W	000Xh	on page 832
FC0A_17F6	(DMA_TCD63_DOFF)	16	R/W	000Xh	on page 833
FC0A_17F8	(DMA_TCD63_DLASTSGA)	32	R/W	0000_000Xh	on page 834
FC0A_17FC	(DMA_TCD63_BITER_ELINKYES)	16	R/W	000Xh	on page 835
	(DMA_TCD63_BITER_ELINKNO)	16	R/W	000Xh	on page 836
FC0A_17FE	(DMA_TCD63_CSR)	16	R/W	000Xh	on page 836

25.3.1 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA.

The DMA arbitrates channel service requests in four groups (0, 1, 2, 3) of 16 channels each. Group 3 contains channels 63–48; group 2 contains channels 47–32; group 1 contains channels 31–16; and group 0 contains channels 15–0.

Arbitration within a group can be configured to use a fixed-priority or a round-robin scheme. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities (see the DCHPRI registers). In Round-Robin Arbitration mode, the channel priorities are ignored, and channels within each group are cycled through without regard to priority.

Note: *Writes to the DMA_CR must be performed only when the DMA channels are inactive (TCDn_CSR[ACTIVE] bits are cleared).*

The group priorities operate in a similar fashion. In Group Fixed-Priority Arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to the occurrence of any channel service requests; otherwise, a configuration error will be reported. In Group Round-Robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Figure 211. Control Register (DMA_CR)

Address: FCOA_0000 + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														CX	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GRP3PRI		GRP2PRI		GRP1PRI		GRP0PRI		EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 291. DMA_CR field descriptions

Field	Description
0–13 Reserved	This read-only field is reserved and always has the value zero.
14 CX	Cancel transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error cancel transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition; thus updating the ES register and generating an optional error interrupt.
16–17 GRP3PRI	Channel group 3 priority Group 3 priority level when fixed-priority group arbitration is enabled.
18–19 GRP2PRI	Channel group 2 priority Group 2 priority level when fixed-priority group arbitration is enabled.
20–21 GRP1PRI	Channel group 1 priority Group 1 priority level when fixed-priority group arbitration is enabled.
22–23 GRP0PRI	Channel group 0 priority Group 0 priority level when fixed-priority group arbitration is enabled.
24 EMLM	Enable minor loop mapping 0 Disabled. TCDn.word2 is defined as a 32-bit NBYTES field 1 Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.

Table 291. DMA_CR field descriptions (continued)

Field	Description
25 CLM	Continuous Link mode 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt on error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable round-robin group arbitration 0 Fixed-priority arbitration is used for selection among the groups. 1 Round-robin arbitration is used for selection among the groups.
29 ERCA	Enable round-robin channel arbitration 0 Fixed-priority arbitration is used for channel selection. 1 Round-robin arbitration is used for channel selection.
30 EDBG	Enable debug 0 When in Debug mode the DMA continues to operate. 1 When in Debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system or the EDBG bit is cleared.
31 Reserved	This read-only bit is reserved and always has the value zero.

25.3.2 Error Status Register (DMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode) or an error termination to a bus master read or write cycle.

See [Section 25.4.3, Error reporting and handling](#), for more details.

Figure 212. Error Status Register (DMA_ES)

Address: FCOA_0000 + 0x0004

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 292. DMA_ES field descriptions

Field	Description
0 VLD	ERRH and ERRL status bits 0 No ERR bits are set 1 At least one ERR bit is set indicating a valid error exists that has not been cleared
1–13 Reserved	This read-only bitfield is reserved and always has the value zero.
14 UCE	Uncorrectable ECC error 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error
15 ECX	Transfer cancelled 0 No cancelled transfers 1 The last recorded entry was a cancelled transfer by the error cancel transfer input
16 GPE	Group priority error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique
17 CPE	Channel priority error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities. Channel priorities are not unique.
18–23 ERRCHN	Error channel number or cancelled channel number GPE and/or last recorded error cancelled transfer
24 SAE	Source address error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE]
25 SOE	Source offset error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].

Table 292. DMA_ES field descriptions (continued)

Field	Description
26 DAE	Destination address error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
27 DOE	Destination offset error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
28 NCE	NBYTES/CITER configuration error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields.
29 SGE	Scatter/gather configuration error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32-byte boundary.
30 SBE	Source bus error 0 No source bus error 1 The last recorded error was a bus error on a source read
31 DBE	Destination bus error 0 No destination bus error 1 The last recorded error was a bus error on a destination write

25.3.3 Enable Request Register High (DMA_ERQH)

The ERQ{H,L} registers provide a bitmap for the 64 implemented channels to enable the request signal for each channel. ERQH supports channels 63–32, while EQRL covers channels 31–00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ{H,L}.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Figure 213. Enable Request Register High (DMA_ERQH)

Address: FCOA_0000 + 0x0008

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 293. DMA_ERQH field descriptions

Field	Description
0 ERQ63	Enable DMA Request 63 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ62	Enable DMA Request 62 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ61	Enable DMA Request 61 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ60	Enable DMA Request 60 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ59	Enable DMA Request 59 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ58	Enable DMA Request 58 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ57	Enable DMA Request 57 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ56	Enable DMA Request 56 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ55	Enable DMA Request 55 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table 293. DMA_ERQH field descriptions (continued)

Field	Description
9 ERQ54	Enable DMA Request 54 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ53	Enable DMA Request 53 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ52	Enable DMA Request 52 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ51	Enable DMA Request 51 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ50	Enable DMA Request 50 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ49	Enable DMA Request 49 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ48	Enable DMA Request 48 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ47	Enable DMA Request 47 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ46	Enable DMA Request 46 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ45	Enable DMA Request 45 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ44	Enable DMA Request 44 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ43	Enable DMA Request 43 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ42	Enable DMA Request 42 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ41	Enable DMA Request 41 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table 293. DMA_ERQH field descriptions (continued)

Field	Description
23 ERQ40	Enable DMA Request 40 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ39	Enable DMA Request 39 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ38	Enable DMA Request 38 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ37	Enable DMA Request 37 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ36	Enable DMA Request 36 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ35	Enable DMA Request 35 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ34	Enable DMA Request 34 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ33	Enable DMA Request 33 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ32	Enable DMA Request 32 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

25.3.4 Enable Request Register Low (DMA_ERQL)

The ERQ{H,L} registers provide a bit map for the 64 implemented channels to enable the request signal for each channel. ERQH supports channels 63–32, while ERQL covers channels 31–00.

The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ{H,L}.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Figure 214. Enable Request Register Low (DMA_ERQL)

Address: FCOA_0000 + 0x000C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ
W	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ	ERQ
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 294. DMA_ERQL field descriptions

Field	Description
0 ERQ31	Enable DMA Request 31 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ28	Enable DMA Request 28 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ25	Enable DMA Request 25 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ23	Enable DMA Request 23 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table 294. DMA_ERQL field descriptions (continued)

Field	Description
9 ERQ22	Enable DMA Request 22 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ21	Enable DMA Request 21 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ16	Enable DMA Request 16 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ15	Enable DMA Request 15 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ11	Enable DMA Request 11 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ10	Enable DMA Request 10 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ9	Enable DMA Request 9 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table 294. DMA_ERQL field descriptions (continued)

Field	Description
23 ERQ8	Enable DMA Request 8 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

25.3.5 Enable Error Interrupt Register High (DMA_EEIH)

The EEI{H,L} registers provide a bit map for the 64 channels to enable the error interrupt signal for each channel. EEIH supports channels 63–32, while EEIL covers channels 31–00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Figure 215. Enable Error Interrupt Register High (DMA_EEIH)

Address: FCOA_0000 + 0x0010

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 295. DMA_EEIH field descriptions

Field	Description
0 EEI63	Enable error interrupt 63 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI62	Enable error interrupt 62 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI61	Enable error interrupt 61 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI60	Enable error interrupt 60 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI59	Enable error interrupt 59 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI58	Enable error interrupt 58 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI57	Enable error interrupt 57 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI56	Enable error interrupt 56 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI55	Enable error interrupt 55 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table 295. DMA_EEIH field descriptions (continued)

Field	Description
9 EEI54	Enable error interrupt 54 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI53	Enable error interrupt 53 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI52	Enable error interrupt 52 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI51	Enable error interrupt 51 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI50	Enable error interrupt 50 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI49	Enable error interrupt 49 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI48	Enable error interrupt 48 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI47	Enable error interrupt 47 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI46	Enable error interrupt 46 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI45	Enable error interrupt 45 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI44	Enable error interrupt 44 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI43	Enable error interrupt 43 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI42	Enable error interrupt 42 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI41	Enable error interrupt 41 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table 295. DMA_EEIH field descriptions (continued)

Field	Description
23 EEI40	Enable error interrupt 40 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI39	Enable error interrupt 39 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI38	Enable error interrupt 38 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI37	Enable error interrupt 37 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI36	Enable error interrupt 36 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI35	Enable error interrupt 35 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI34	Enable error interrupt 34 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI33	Enable error interrupt 33 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI32	Enable error interrupt 32 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

25.3.6 Enable Error Interrupt Register Low (DMA_EEIL)

The Enable Error Interrupt Register Low (DMA_EEIL) provides a bit map for the 64 channels to enable the error interrupt signal for each channel. EEIH supports channels 63–32, while EEIL covers channels 31–00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Figure 216. Enable Error Interrupt Register Low (DMA_EEIL)

Address: FCOA_0000 + 0x0014

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EEI 31	EEI 30	EEI 29	EEI 28	EEI 27	EEI 26	EEI 25	EEI 24	EEI 23	EEI 22	EEI 21	EEI 20	EEI 19	EEI 18	EEI 17	EEI 16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI 15	EEI 14	EEI 13	EEI 12	EEI 11	EEI 10	EEI 9	EEI 8	EEI 7	EEI 6	EEI 5	EEI 4	EEI 3	EEI 2	EEI 1	EEI 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 296. DMA_EEIL field descriptions

Field	Description
0 EEI31	Enable error interrupt 31 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI30	Enable error interrupt 30 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable error interrupt 29 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable error interrupt 28 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI27	Enable error interrupt 27 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable error interrupt 26 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable error interrupt 25 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable error interrupt 24 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI23	Enable error interrupt 23 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table 296. DMA_EEIL field descriptions (continued)

Field	Description
9 EEI22	Enable error interrupt 22 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable error interrupt 21 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable error interrupt 20 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI19	Enable error interrupt 19 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable error interrupt 18 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable error interrupt 17 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable error interrupt 16 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI15	Enable error interrupt 15 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable error interrupt 14 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI13	Enable error interrupt 13 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI12	Enable error interrupt 12 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI11	Enable error interrupt 11 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable error interrupt 10 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable error interrupt 9 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table 296. DMA_EEIL field descriptions (continued)

Field	Description
23 EEI8	Enable error interrupt 8 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI7	Enable error interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable error interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable error interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable error interrupt 4 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI3	Enable error interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable error interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI1	Enable error interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable error interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

25.3.7 Set Enable Request Register (DMA_SERQ)

The Set Enable Request Register (DMA_SERQ) provides a simple memory-mapped mechanism to set a given bit in the ERQ{H,L} to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ{H,L} to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ{H,L} to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Figure 217. Set Enable Request Register (DMA_SERQ)

Address: FCOA_0000 + 0x0018

Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAER	SERQ					
Reset	0	0	0	0	0	0	0	0

Table 297. DMA_SERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAER	Set all enable requests 0 Set only those ERQ bits specified in the SERQ field 1 Set all bits in ERQ{H,L}
2–7 SERQ	Set enable request Sets the corresponding bit in ERQ{H,L}

25.3.8 Clear Enable Request Register (DMA_CERQ)

The Clear Enable Request Register (DMA_CERQ) provides a simple memory-mapped mechanism to clear a given bit in the ERQ{H,L} to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ{H,L} to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ{H,L} to be cleared, disabling all DMA request inputs.

If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Figure 218. Clear Enable Request Register (DMA_CERQ)

Address: FCOA_0000 + 0x0019

Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAER	CERQ					
Reset	0	0	0	0	0	0	0	0

Table 298. DMA_CERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register

Table 298. DMA_CERQ field descriptions (continued)

Field	Description
1 CAER	Clear all enable requests 0 Clear only those ERQ bits specified in the CERQ field 1 Clear all bits in ERQ
2–7 CERQ	Clear enable request Clears the corresponding bit in ERQ{H,L}

25.3.9 Set Enable Error Interrupt Register (DMA_SEEI)

The Set Enable Error Interrupt Register (DMA_SEEI) provides a simple memory-mapped mechanism to set a given bit in the EEI{H,L} to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI{H,L} to be set. Setting the SAAE bit provides a global set function, forcing the entire EEI{H,L} contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Figure 219. Set Enable Error Interrupt Register (DMA_SEEI)

Address: FCOA_0000 + 0x001A

Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAAE	SEEI					
Reset	0	0	0	0	0	0	0	0

Table 299. DMA_SEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAAE	Sets all enable error interrupts 0 Set only those EEI bits specified in the SEEI field. 1 Sets all bits in EEI
2–7 SEEI	Set enable error interrupt Sets the corresponding bit in EEI{H,L}

25.3.10 Clear Enable Error Interrupt Register (DMA_CEEI)

The Clear Enable Error Interrupt Register (DMA_CEEI) provides a simple memory-mapped mechanism to clear a given bit in the EEI{H,L} to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI{H,L} to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI{H,L} contents to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Figure 220. Clear Enable Error Interrupt Register (DMA_CEEI)

Address: FC0A_0000 + 0x001B

Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAEE	CEEI					
Reset	0	0	0	0	0	0	0	0

Table 300. DMA_CEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAEE	Clear all enable error interrupts 0 Clear only those EEI bits specified in the CEEI field 1 Clear all bits in EEI
2–7 CEEI	Clear enable error interrupt Clears the corresponding bit in EEI{H,L}

25.3.11 Clear Interrupt Request Register (DMA_CINT)

The Clear Interrupt Request Register (DMA_CINT) provides a simple, memory-mapped mechanism to clear a given bit in the INT{H,L} to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT{H,L} to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT{H,L} to be cleared, disabling all DMA interrupt requests.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Figure 221. Clear Interrupt Request Register (DMA_CINT)

Address: FC0A_0000 + 0x001C

Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAIR	CINT					
Reset	0	0	0	0	0	0	0	0

Table 301. DMA_CINT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register

Table 301. DMA_CINT field descriptions (continued)

Field	Description
1 CAIR	Clear all interrupt requests 0 Clear only those INT bits specified in the CINT field 1 Clear all bits in INT{H,L}
2–7 CINT	Clear interrupt request Clears the corresponding bit in INT{H,L}

25.3.12 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR{H,L} to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR{H,L} to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR{H,L} contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Figure 222. Clear Error Register (DMA_CERR)

Address: FC0A_0000 + 0x001D

Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAEI	CERR					
Reset	0	0	0	0	0	0	0	0

Table 302. DMA_CERR field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAEI	Clear all error indicators 0 Clear only those ERR bits specified in the CERR field 1 Clear all bits in ERR
2–7 CERR	Clear error indicator Clears the corresponding bit in ERR{H,L}

25.3.13 Set START Bit Register (DMA_SSRT)

The Set START Bit Register (DMA_SSRT) provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Figure 223. Set START Bit Register (DMA_SSRT)

Address: FC0A_0000 + 0x001E

Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAST	SSRT					
Reset	0	0	0	0	0	0	0	0

Table 303. DMA_SSRT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAST	Set all START bits (activates all channels) 0 Set only those TCD _n _CSR[START] bits specified in the SSRT field 1 Set all bits in TCD _n _CSR[START]
2–7 SSRT	Set START bit Sets the corresponding bit in TCD _n _CSR[START]

25.3.14 Clear DONE Status Bit Register (DMA_CDNE)

The Clear DONE Status Bit Register (DMA_CDNE) provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Figure 224. Clear DONE Status Bit Register (DMA_CDNE)

Address: FC0A_0000 + 0x001F

Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CADN	CDNE					
Reset	0	0	0	0	0	0	0	0

Table 304. DMA_CDNE field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register

Table 304. DMA_CDNE field descriptions (continued)

Field	Description
1 CADN	Clears all DONE bits 0 Clears only those TCDn_CSR[DONE] bits specified in the CDNE field 1 Clears all bits in TCDn_CSR[DONE]
2–7 CDNE	Clear DONE bit Clears the corresponding bit in TCDn_CSR[DONE]

25.3.15 Interrupt Request Register High (DMA_INTH)

The Interrupt Request Register High (DMA_INTH) provides a bit map for the upper half of 64 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptions, the eDMA engine generates an interrupt on a data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT. On writes to the INT, a '1' in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The CINT is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L}.

Figure 225. Interrupt Request Register High (DMA_INTH)

Address: FC0A_0000 + 0x0020

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT 63	INT 62	INT 61	INT 60	INT 59	INT 58	INT 57	INT 56	INT 55	INT 54	INT 53	INT 52	INT 51	INT 50	INT 49	INT 48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT 47	INT 46	INT 45	INT 44	INT 43	INT 42	INT 41	INT 40	INT 39	INT 38	INT 37	INT 36	INT 35	INT 34	INT 33	INT 32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 305. DMA_INTH field descriptions

Field	Description
0 INT63	Interrupt request 63 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT62	Interrupt request 62 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT61	Interrupt request 61 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT60	Interrupt request 60 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT59	Interrupt request 59 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT58	Interrupt request 58 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT57	Interrupt request 57 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT56	Interrupt request 56 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT55	Interrupt request 55 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT54	Interrupt request 54 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT53	Interrupt request 53 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT52	Interrupt request 52 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT51	Interrupt request 51 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT50	Interrupt request 50 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

Table 305. DMA_INTH field descriptions (continued)

Field	Description
14 INT49	Interrupt request 49 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT48	Interrupt request 48 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT47	Interrupt request 47 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT46	Interrupt request 46 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT45	Interrupt request 45 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT44	Interrupt request 44 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT43	Interrupt request 43 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT42	Interrupt request 42 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT41	Interrupt request 41 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT40	Interrupt request 40 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT39	Interrupt request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT38	Interrupt request 38 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT37	Interrupt request 37 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT36	Interrupt request 36 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

Table 305. DMA_INTH field descriptions (continued)

Field	Description
28 INT35	Interrupt request 35 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT34	Interrupt request 34 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT33	Interrupt request 33 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT32	Interrupt request 32 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

25.3.16 Interrupt Request Register Low (DMA_INTL)

The Interrupt Request Register Low (DMA_INTL) provides a bit map for the lower half of the 64 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt at data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT. On writes to the INT, a '1' in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The CINT is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L}.

Figure 226. Interrupt Request Register Low (DMA_INTL)

Address: FCOA_0000 + 0x0024

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT 31	INT 30	INT 29	INT 28	INT 27	INT 26	INT 25	INT 24	INT 23	INT 22	INT 21	INT 20	INT 19	INT 18	INT 17	INT 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT 15	INT 14	INT 13	INT 12	INT 11	INT 10	INT 9	INT 8	INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 306. DMA_INTL field descriptions

Field	Description
0 INT31	Interrupt request 31 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT30	Interrupt request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT29	Interrupt request 29 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT28	Interrupt request 28 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT27	Interrupt request 27 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT26	Interrupt request 26 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT25	Interrupt request 25 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT24	Interrupt request 24 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT23	Interrupt request 23 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

Table 306. DMA_INTL field descriptions (continued)

Field	Description
9 INT22	Interrupt request 22 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT21	Interrupt request 21 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT20	Interrupt request 20 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT19	Interrupt request 19 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT18	Interrupt request 18 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT17	Interrupt request 17 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT16	Interrupt request 16 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT15	Interrupt request 15 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT14	Interrupt request 14 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT13	Interrupt request 13 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT12	Interrupt request 12 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT11	Interrupt request 11 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT10	Interrupt request 10 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT9	Interrupt request 9 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

Table 306. DMA_INTL field descriptions (continued)

Field	Description
23 INT8	Interrupt request 8 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT7	Interrupt request 7 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT6	Interrupt request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT5	Interrupt request 5 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT4	Interrupt request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT3	Interrupt request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT2	Interrupt request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT1	Interrupt request 1 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT0	Interrupt request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

25.3.17 Error Register High (DMA_ERRH)

The ERR{H,L} provides a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63–32, while ERRL covers channels 31–00. The eDMA engine signals the occurrence of a error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a nonzero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding

channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Figure 227. Error Register High (DMA_ERRH)

Address: FC0A_0000 + 0x0028

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR 63	ERR 62	ERR 61	ERR 60	ERR 59	ERR 58	ERR 57	ERR 56	ERR 55	ERR 54	ERR 53	ERR 52	ERR 51	ERR 50	ERR 49	ERR 48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR 47	ERR 46	ERR 45	ERR 44	ERR 43	ERR 42	ERR 41	ERR 40	ERR 39	ERR 38	ERR 37	ERR 36	ERR 35	ERR 34	ERR 33	ERR 32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 307. DMA_ERRH field descriptions

Field	Description
0 ERR63	Error in channel 63 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
1 ERR62	Error in channel 62 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
2 ERR61	Error in channel 61 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
3 ERR60	Error in channel 60 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
4 ERR59	Error in channel 59 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
5 ERR58	Error in channel 58 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
6 ERR57	Error in channel 57 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table 307. DMA_ERRH field descriptions (continued)

Field	Description
7 ERR56	Error in channel 56 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
8 ERR55	Error in channel 55 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
9 ERR54	Error in channel 54 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
10 ERR53	Error in channel 53 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
11 ERR52	Error in channel 52 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
12 ERR51	Error in channel 51 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
13 ERR50	Error in channel 50 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
14 ERR49	Error in channel 49 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
15 ERR48	Error in channel 48 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
16 ERR47	Error in channel 47 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
17 ERR46	Error in channel 46 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
18 ERR45	Error in channel 45 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
19 ERR44	Error in channel 44 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
20 ERR43	Error in channel 43 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table 307. DMA_ERRH field descriptions (continued)

Field	Description
21 ERR42	Error in channel 42 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
22 ERR41	Error in channel 41 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
23 ERR40	Error in channel 40 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
24 ERR39	Error in channel 39 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
25 ERR38	Error in channel 38 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
26 ERR37	Error in channel 37 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
27 ERR36	Error in channel 36 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
28 ERR35	Error in channel 35 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
29 ERR34	Error in channel 34 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
30 ERR33	Error in channel 33 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
31 ERR32	Error in channel 32 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

25.3.18 Error Register Low (DMA_ERRL)

The ERR{H,L} provides a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63–32, while ERRL covers channels 31–00. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel

completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Figure 228. Error Register Low (DMA_ERRL)

Address: FC0A_0000 + 0x002C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR 31	ERR 30	ERR 29	ERR 28	ERR 27	ERR 26	ERR 25	ERR 24	ERR 23	ERR 22	ERR 21	ERR 20	ERR 19	ERR 18	ERR 17	ERR 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR 15	ERR 14	ERR 13	ERR 12	ERR 11	ERR 10	ERR 9	ERR 8	ERR 7	ERR 6	ERR 5	ERR 4	ERR 3	ERR 2	ERR 1	ERR 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 308. DMA_ERRL field descriptions

Field	Description
0 ERR31	Error in channel 31 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
1 ERR30	Error in channel 30 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
2 ERR29	Error in channel 29 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
3 ERR28	Error in channel 28 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
4 ERR27	Error in channel 27 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
5 ERR26	Error in channel 26 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table 308. DMA_ERRL field descriptions (continued)

Field	Description
6 ERR25	Error in channel 25 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
7 ERR24	Error in channel 24 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
8 ERR23	Error in channel 23 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
9 ERR22	Error in channel 22 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
10 ERR21	Error in channel 21 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
11 ERR20	Error in channel 20 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
12 ERR19	Error in channel 19 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
13 ERR18	Error in channel 18 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
14 ERR17	Error in channel 17 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
15 ERR16	Error in channel 16 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
16 ERR15	Error in channel 15 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
17 ERR14	Error in channel 14 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
18 ERR13	Error in channel 13 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
19 ERR12	Error in channel 12 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table 308. DMA_ERRL field descriptions (continued)

Field	Description
20 ERR11	Error in channel 11 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
21 ERR10	Error in channel 10 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
22 ERR9	Error in channel 9 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
23 ERR8	Error in channel 8 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
24 ERR7	Error in channel 7 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
25 ERR6	Error in channel 6 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
26 ERR5	Error in channel 5 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
27 ERR4	Error in channel 4 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
28 ERR3	Error in channel 3 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
29 ERR2	Error in channel 2 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
30 ERR1	Error in channel 1 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
31 ERR0	Error in channel 0 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

25.3.19 Hardware Request Status Register High (DMA_HRSH)

The HRS{H,L} provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. HRSH supports channels 63–32, while HRSL covers channels 31–00. The Hardware Request Status bits reflect the current state of the register and qualified (via the ERQ{H,L} fields) DMA request signals as seen by the

DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

Note: *Note: These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ{H,L} bits.*

Figure 229. Hardware Request Status Register High (DMA_HRSH)

Address: FCOA_0000 + 0x0030

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 309. DMA_HRSH field descriptions

Field	Description
0 HRS63	Hardware request status channel 63 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
1 HRS62	Hardware request status channel 62 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
2 HRS61	Hardware request status channel 61 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
3 HRS60	Hardware request status channel 60 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
4 HRS59	Hardware request status channel 59 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
5 HRS58	Hardware request status channel 58 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
6 HRS57	Hardware request status channel 57 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
7 HRS56	Hardware request status channel 56 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table 309. DMA_HRSH field descriptions (continued)

Field	Description
8 HRS55	Hardware request status channel 55 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
9 HRS54	Hardware request status channel 54 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
10 HRS53	Hardware request status channel 53 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
11 HRS52	Hardware request status channel 52 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
12 HRS51	Hardware request status channel 51 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
13 HRS50	Hardware request status channel 50 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
14 HRS49	Hardware request status channel 49 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
15 HRS48	Hardware request status channel 48 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
16 HRS47	Hardware request status channel 47 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
17 HRS46	Hardware request status channel 46 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
18 HRS45	Hardware request status channel 45 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
19 HRS44	Hardware request status channel 44 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
20 HRS43	Hardware request status channel 43 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
21 HRS42	Hardware request status channel 42 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table 309. DMA_HRSH field descriptions (continued)

Field	Description
22 HRS41	Hardware request status channel 41 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
23 HRS40	Hardware request status channel 40 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
24 HRS39	Hardware request status channel 39 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
25 HRS38	Hardware request status channel 38 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
26 HRS37	Hardware request status channel 37 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
27 HRS36	Hardware request status channel 36 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
28 HRS35	Hardware request status channel 35 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
29 HRS34	Hardware request status channel 34 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
30 HRS33	Hardware request status channel 33 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
31 HRS32	Hardware request status channel 32 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

25.3.20 Hardware Request Status Register Low (DMA_HRSL)

The HRS{H,L} provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. HRSH supports channels 63–32, while HRSL covers channels 31–00. The Hardware Request Status bits reflect the current state of the register and qualified (via the ERQ{H,L} fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

Note: These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ{H,L} bits.

Figure 230. Hardware Request Status Register Low (DMA_HRSL)

Address: FCOA_0000 + 0x0034

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS
W	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 310. DMA_HRSL field descriptions

Field	Description
0 HRS31	Hardware request status channel 31 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
1 HRS30	Hardware request status channel 30 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
2 HRS29	Hardware request status channel 29 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
3 HRS28	Hardware request status channel 28 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
4 HRS27	Hardware request status channel 27 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
5 HRS26	Hardware request status channel 26 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
6 HRS25	Hardware request status channel 25 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
7 HRS24	Hardware request status channel 24 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
8 HRS23	Hardware request status channel 23 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table 310. DMA_HRSL field descriptions (continued)

Field	Description
9 HRS22	Hardware request status channel 22 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
10 HRS21	Hardware request status channel 21 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
11 HRS20	Hardware request status channel 20 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
12 HRS19	Hardware request status channel 19 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
13 HRS18	Hardware request status channel 18 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
14 HRS17	Hardware request status channel 17 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
15 HRS16	Hardware request status channel 16 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
16 HRS15	Hardware request status channel 15 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
17 HRS14	Hardware request status channel 14 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
18 HRS13	Hardware request status channel 13 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
19 HRS12	Hardware request status channel 12 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
20 HRS11	Hardware request status channel 11 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
21 HRS10	Hardware request status channel 10 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
22 HRS9	Hardware request status channel 9 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table 310. DMA_HRSL field descriptions (continued)

Field	Description
23 HRS8	Hardware request status channel 8 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
24 HRS7	Hardware request status channel 7 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
25 HRS6	Hardware request status channel 6 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
26 HRS5	Hardware request status channel 5 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
27 HRS4	Hardware request status channel 4 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
28 HRS3	Hardware request status channel 3 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
29 HRS2	Hardware request status channel 2 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
30 HRS1	Hardware request status channel 1 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
31 HRS0	Hardware request status channel 0 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

25.3.21 Channel *n* Priority Register (DMA_DCHPRIn)

When the Fixed-Priority Channel Arbitration mode is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values. Otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15.

When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMA control register.

Figure 231. Channel n Priority Register (DMA_DCHPRIn)Address: FCOA_0000h base + 100h offset + (1d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7
R	ECP	DPA	GRPPRI		CHPRI			
W								
Reset ⁽¹⁾	0	0	x	x	x	x	x	x

1. x = Undefined at reset.

Table 311. DMA_DCHPRIn field descriptions

Field	Description
0 ECP	Enable channel preemption This bit resets to zero. 0 Channel n cannot be suspended by a higher priority channel's service request 1 Channel n can be temporarily suspended by the service request of a higher priority channel
1 DPA	Disable preempt ability This bit resets to zero. 0 Channel n can suspend a lower priority channel 1 Channel n cannot suspend any channel, regardless of channel priority
2–3 GRPPRI	Channel n current group priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. <i>Note: Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, i.e., DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.</i>
4–7 CHPRI	Channel n arbitration priority Channel priority when fixed-priority arbitration is enabled <i>Note: Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, i.e., DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.</i>

25.3.22 Channel n Master ID Register (DMA_DCHMIDn)

The DMA Master ID Replication registers allow the DMA to use the same protection level and AHB system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master (a core, for example) programs a TCD, its master ID and protection level are captured when the TCD Word 7 control attributes are written. Although the scatter/gather operation can change the contents of TCD Word 7, that operation does not affect the DCHMID n registers.

Figure 232. Channel *n* Master ID Register (DMA_DCHMIDn)Address: FC0A_0000h base + 140h offset + (1d × *n*), where *n* = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7
R	EMI	PAL	0		MID			
W								
Reset	0	0	0	0	0	0	0	0

Table 312. DMA_DCHMIDn field descriptions

Field	Description
0 EMI	Enable Master ID replication 0 Master ID replication is disabled 1 Master ID replication is enabled
1 PAL	Privileged Access Level The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCD Word 7. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
2–3 Reserved	This read-only bitfield is reserved and always has the value zero.
4–7MID	Master ID DMA's master ID when channel <i>n</i> is active and master ID replication is enabled. <i>Note: The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCD Word 7.</i>

25.3.23 TCD Source Address (DMA_TCDn_SADDR)

See TCD Source Address register figure and DMA_TCDn_SADDR field descriptions table as follows.

Figure 233. TCD Source Address (DMA_TCDn_SADDR)Address: FC0A_0000h base + 1000h offset + (32d × *n*), where *n* = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SADDR															
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADDR															
W																
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 313. DMA_TCDn_SADDR field descriptions

Field	Description
0–31 SADDR	Source address Memory address pointing to the source data.

25.3.24 TCD Transfer Attributes (DMA_TCDn_ATTR)

See TCD Transfer Attributes register figure and DMA_TCDn_ATTR field descriptions table as follows.

Figure 234. TCD Transfer Attributes (DMA_TCDn_ATTR)

Address: FC0A_0000h base + 1004h offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMOD				SSIZE				DMOD				DSIZE			
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 314. DMA_TCDn_ATTR field descriptions

Field	Description
16–20 SMOD	Source address modulo. 0 Source address modulo feature is disabled ≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
21–23 SSIZE	Source data transfer size Using a reserved encoding causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte burst 110 Reserved 111 Reserved

Table 314. DMA_TCDn_ATTR field descriptions (continued)

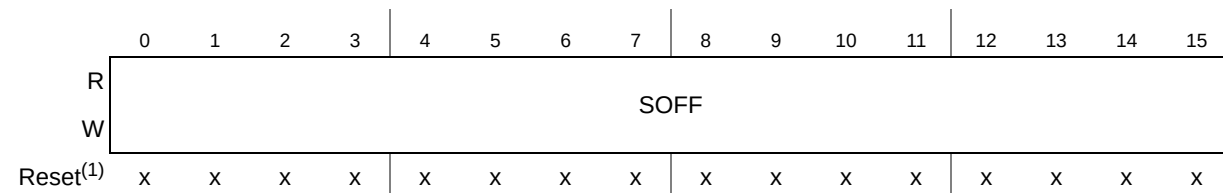
Field	Description
24–28 DMOD	Destination address modulo See the SMOD definition
29–31 DSIZE	Destination data transfer size See the SSIZE definition

25.3.25 TCD Signed Source Address Offset (DMA_TCDn_SOFF)

See TCD Signed Source Address Offset register figure and DMA_TCDn_SOFF field descriptions table as follows.

Figure 235. TCD Signed Source Address Offset (DMA_TCDn_SOFF)

Address: FC0A_0000h base + 1006h offset + (32d × n), where n = 0d to 63d Access: User read/write



1. x = Undefined at reset.

Table 315. DMA_TCDn_SOFF field descriptions

Field	Description
16–31 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

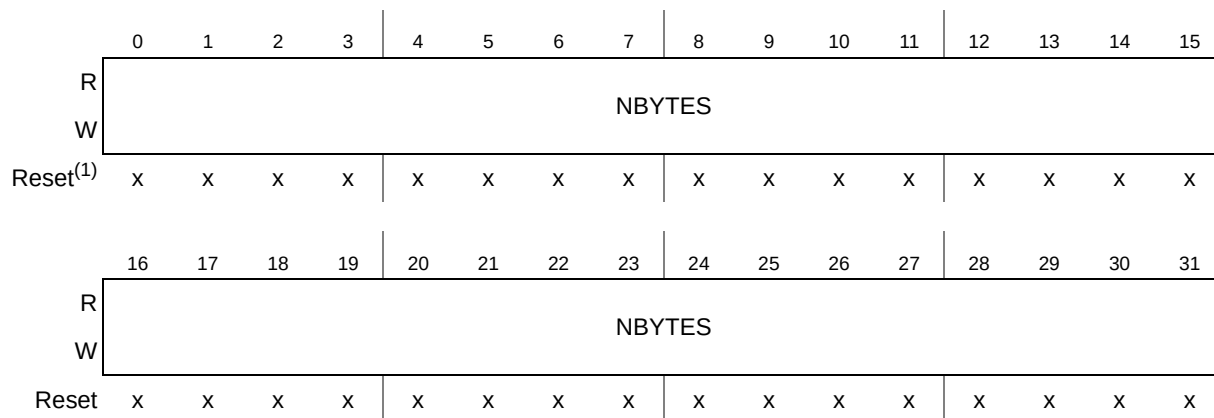
25.3.26 TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCDn_NBYTES_MLNO)

See TCD Minor Byte Count (Minor Loop Disabled) register figure and DMA_TCDn_NBYTES_MLNO field descriptions table below.

If minor loop mapping is disabled (CR[EMLM] = 0), TCD word 2 is defined as follows.

Figure 236. TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCDn_NBYTES_MLNO)

Address: FC0A_0000h base + 1008h offset + (32d × n), where n = 0d to 63d Access: User read/write



1. x = Undefined at reset.

Table 316. DMA_TCDn_NBYTES_MLNO field descriptions

Field	Description
0–31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. <i>Note: An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.</i>

25.3.27 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCDn_NBYTES_MLOFFNO)

TCD Word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

**Figure 237. TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled)
(DMA_TCDn_NBYTES_MLOFFNO)**

Address: FC0A_0000h base + 1008h offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMLOE	DMLOE	NBYTES													
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NBYTES															
W																
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

25.3.28 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCDn_NBYTES_MLOFFYES)

TCD Word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset enabled (SMLOE or DMLOE = 1)

**Figure 238. TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled)
(DMA_TCDn_NBYTES_MLOFFYES)**

Address: FC0A_0000h base + 1008h offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMLOE	DMLOE	MLOFF													
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MLOFF								NBYTES							
W																
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

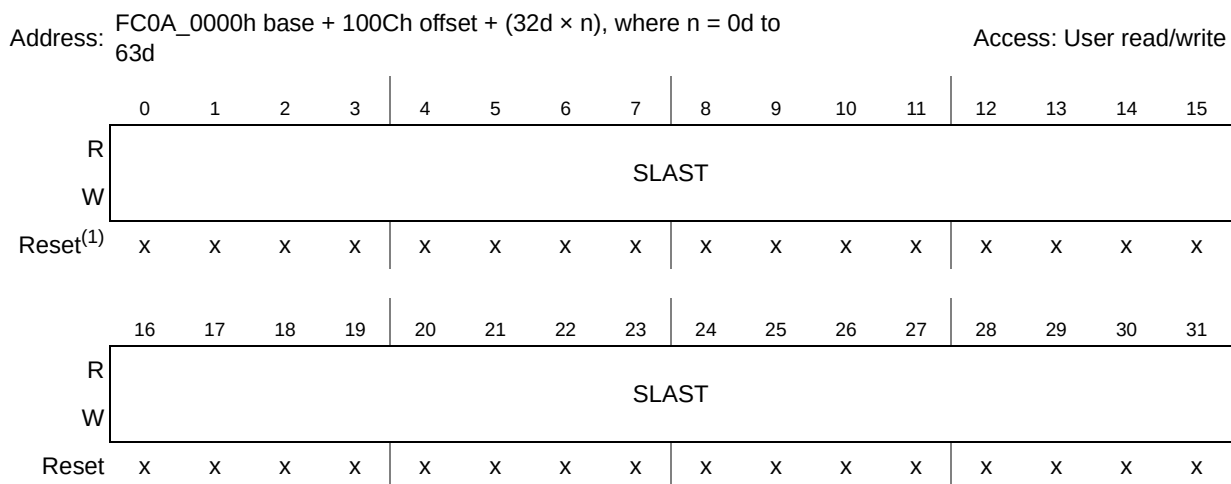
1. x = Undefined at reset.

Table 317. DMA_TCDn_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
2–21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22–31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

25.3.29 TCD Last Source Address Adjustment (DMA_TCDn_SLAST)

Figure 239. TCD Last Source Address Adjustment (DMA_TCDn_SLAST)



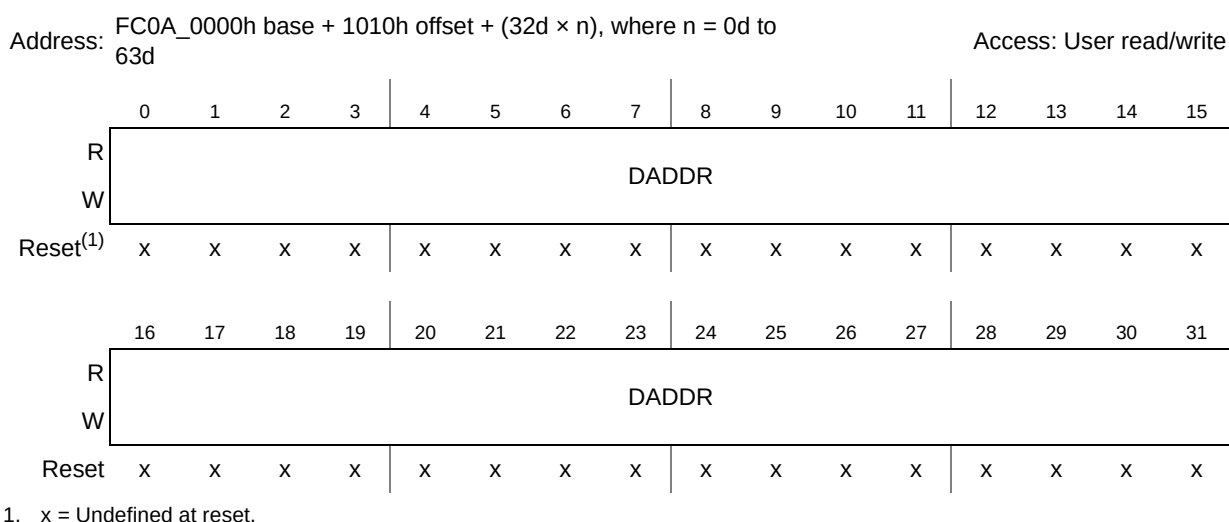
1. x = Undefined at reset.

Table 318. DMA_TCDn_SLAST field descriptions

Field	Description
0–31 SLAST	Last source address adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.

25.3.30 TCD Destination Address (DMA_TCDn_DADDR)

See TCD Destination Address register figure and DMA_TCDn_DADDR field descriptions table as follows.

Figure 240. TCD Destination Address (DMA_TCDn_DADDR)**Table 319. DMA_TCDn_DADDR field descriptions**

Field	Description
0–31 DADDR	Destination address Memory address pointing to the destination data.

25.3.31 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_CITER_ELINKYES)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) register figure and DMA_TCDn_CITER_ELINKYES field descriptions table as follows.

**Figure 241. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled)
(DMA_TCDn_CITER_ELINKYES)**

Address: FCOA_0000h base + 1014h offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	E	LINKCH						CITER								
W	LINK															
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 320. DMA_TCDn_CITER_ELINKYES field descriptions

Field	Description
16 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. <i>Note: This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported.</i> 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
17–22 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by these six bits by setting that channel's TCDn_CSR[START] bit.
23–31 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (e.g., final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. <i>Note: When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.</i>

25.3.32 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_CITER_ELINKNO)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) register figure and DMA_TCDn_CITER_ELINKNO field descriptions table as follows.

**Figure 242. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled)
(DMA_TCDn_CITER_ELINKNO)**

Address: FC0A_0000h base + 1014h offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELINK	CITER														
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 321. DMA_TCDn_CITER_ELINKNO field descriptions

Field	Description
16 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. <i>Note: This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported.</i> 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
17–31 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (for example, final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. <i>Note: When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.</i>

25.3.33 TCD Signed Destination Address Offset (DMA_TCDn_DOFF)

See TCD Signed Destination Address Offset register figure and DMA_TCDn_DOFF field descriptions table as follows.

Figure 243. TCD Signed Destination Address Offset (DMA_TCDn_DOFF)

Address: FC0A_0000h base + 1016h offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DOFF															
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 322. DMA_TCDn_DOFF field descriptions

Field	Description
16–31 DOFF	Destination address signed offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

25.3.34 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCDn_DLASTSGA)

Figure 244. TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCDn_DLASTSGA)

Address: FC0A_0000h base + 1018h offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DLASTSGA															
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DLASTSGA															
W																
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 323. DMA_TCDn_DLASTSGA field descriptions

Field	Description
0–31 DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then – Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. else – This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, else a configuration error is reported.

25.3.35 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_BITER_ELINKYES)

Figure 245. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_BITER_ELINKYES)

Address: FC0A_0000h base + 101Ch offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELINK	LINKCH						BITER								
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 324. DMA_TCDn_BITER_ELINKYES field descriptions

Field	Description
16 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. <i>Note: When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.</i> 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
17–22 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCDn_CSR[START] bit. <i>Note: When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.</i>
23–31 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. <i>Note: When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.</i>

25.3.36 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_BITER_ELINKNO)

Figure 246. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_BITER_ELINKNO)

Address: FCOA_0000h base + 101Ch offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELINK	BITER														
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 325. DMA_TCDn_BITER_ELINKNO field descriptions

Field	Description
16 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. <i>Note: When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.</i> 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
17–31 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. <i>Note: When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.</i>

25.3.37 TCD Control and Status (DMA_TCDn_CSR)

See TCD Control and Status register figure and DMA_TCDn_CSR field descriptions table as follows.

Figure 247. TCD Control and Status (DMA_TCDn_CSR)

Address: FC0A_0000h base + 101Eh offset + (32d × n), where n = 0d to 63d

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BWC		MAJORLINKCH						DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
W																
Reset ⁽¹⁾	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Table 326. DMA_TCDn_CSR field descriptions

Field	Description
16–17 BWC	Bandwidth control Throttles the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch (XBS). <i>Note: If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency.</i> 00 No eDMA engine stalls 01 Reserved 10 eDMA engine stalls for 4 cycles after each r/w 11 eDMA engine stalls for 8 cycles after each r/w
18–23 MAJORLINKCH	Link channel number If (MAJORELINK = 0) then – No channel-to-channel linking (or chaining) is performed after the major loop counter is exhausted. else – After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCDn_CSR[START] bit.
24 DONE	Channel done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero; The software clears it, or the hardware when the channel is activated. <i>Note: This bit must be cleared to write the MAJORELINK or ESG bits.</i> This bit resets to zero.
25 ACTIVE	Channel active This flag signals the channel is currently in execution. It is set when channel service begins, and the eDMA clears it as the minor loop completes or if any error condition is detected.

Table 326. DMA_TCDn_CSR field descriptions (continued)

Field	Description
26 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. <i>Note: To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.</i> 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
27 ESG	Enable scatter/gather processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. <i>Note: To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.</i> 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
28 DREQ	Disable request {H,L} 0 The channel's ERQ bit is not affected 1 The channel's ERQ bit is cleared when the major loop is complete
29 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered (aka ping-pong) schemes or other types of data movement where the processor needs an early indication of the transfer's progress. <i>Note: If BITER is set, do not use INTHALF. Use INTMAJOR instead.</i> 0 The half-point interrupt is disabled 1 The half-point interrupt is enabled
30 INTMAJOR	Enable an interrupt when major iteration count completes If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled 1 The end-of-major loop interrupt is enabled
31 START	Channel start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. This bit resets to zero. 0 The channel is not explicitly started 1 The channel is explicitly started via a software initiated service request

25.4 Functional description

This section provides a description of the DMA request sources, an overview of the microarchitecture, and functional operation of the eDMA module.

25.4.1 eDMA microarchitecture

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules:

- eDMA engine
 - Address path:

This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRIn[ECPI]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the $TCDn_SADDR$, $TCDn_DADDR$, $TCDn_CITER$ back to local memory. If the major iteration count is exhausted, additional processing is performed, including updates to the final address pointer, reloading the $TCDn_CITER$ field, and a possible fetch of the next $TCDn$ from memory as part of a scatter/gather operation.
 - Data path:

This block implements the bus master read/write datapath. It includes 32 bytes of register storage and the necessary multiplex logic to support any data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output.

The address and data path modules directly support the two-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase), while the data path module implements the second stage of the pipeline (data phase).
 - Program model/channel arbitration:

This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus (not shown). The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
 - Control:

This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-

bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

- Transfer control descriptor memory
 - Memory controller:

This logic implements the dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus.

As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
 - Memory array:

TCD storage is implemented using a single-port, synchronous RAM array.

25.4.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments. As shown in [Figure 248](#), the first segment involves the channel activation. In the diagram, this example uses the assertion of the eDMA peripheral request signal to request service for channel n . Channel activation via software and the $TCDn_CSR[START]$ bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, channel arbitration is performed, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the address to access the local memory for $TCDn$. Next, the TCD memory is accessed and the descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

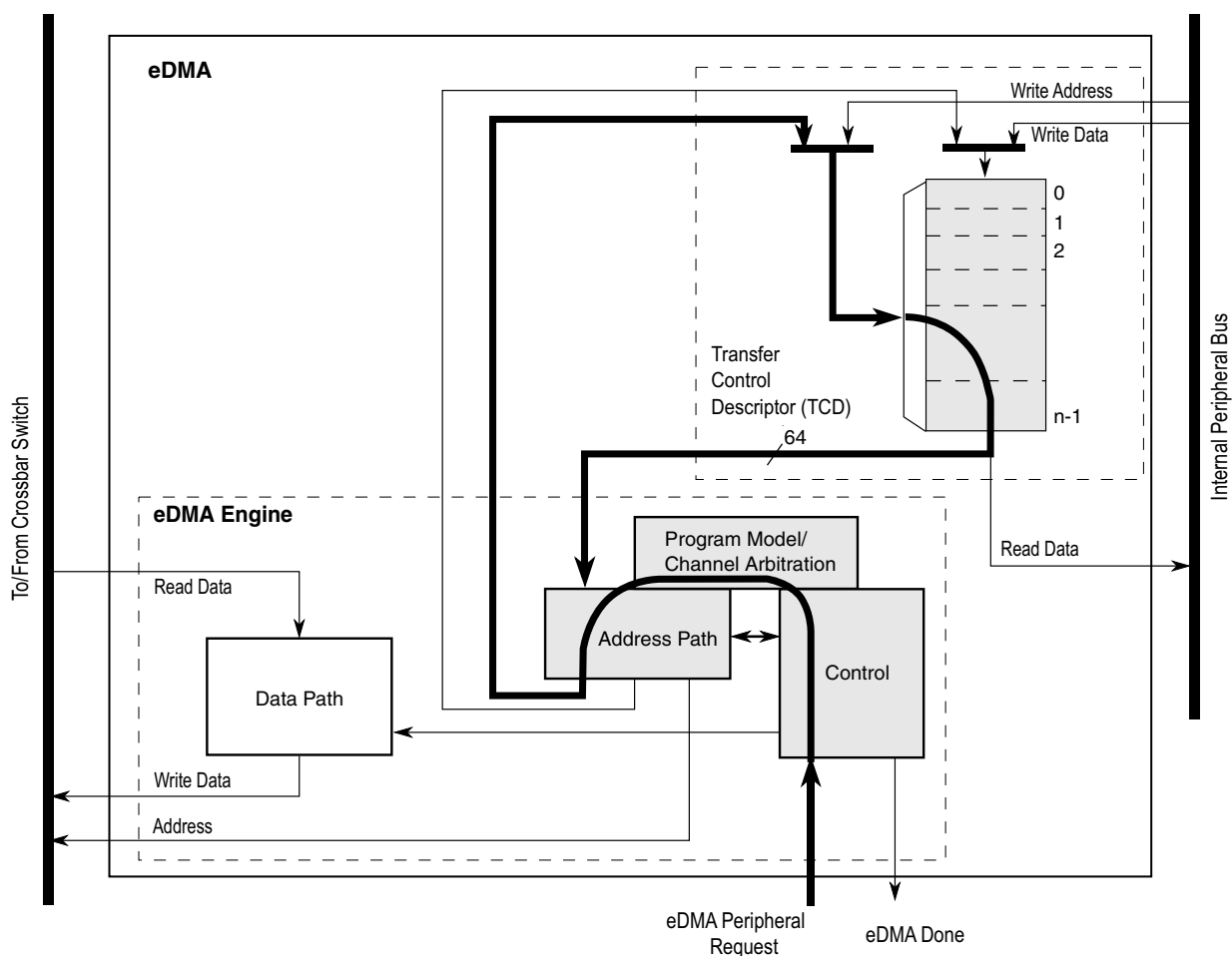


Figure 248. DMA operation, part 1

In the second part of the basic data flow as shown in [Figure 249](#) the modules associated with the data transfer (address path, data path, and control) sequence through source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

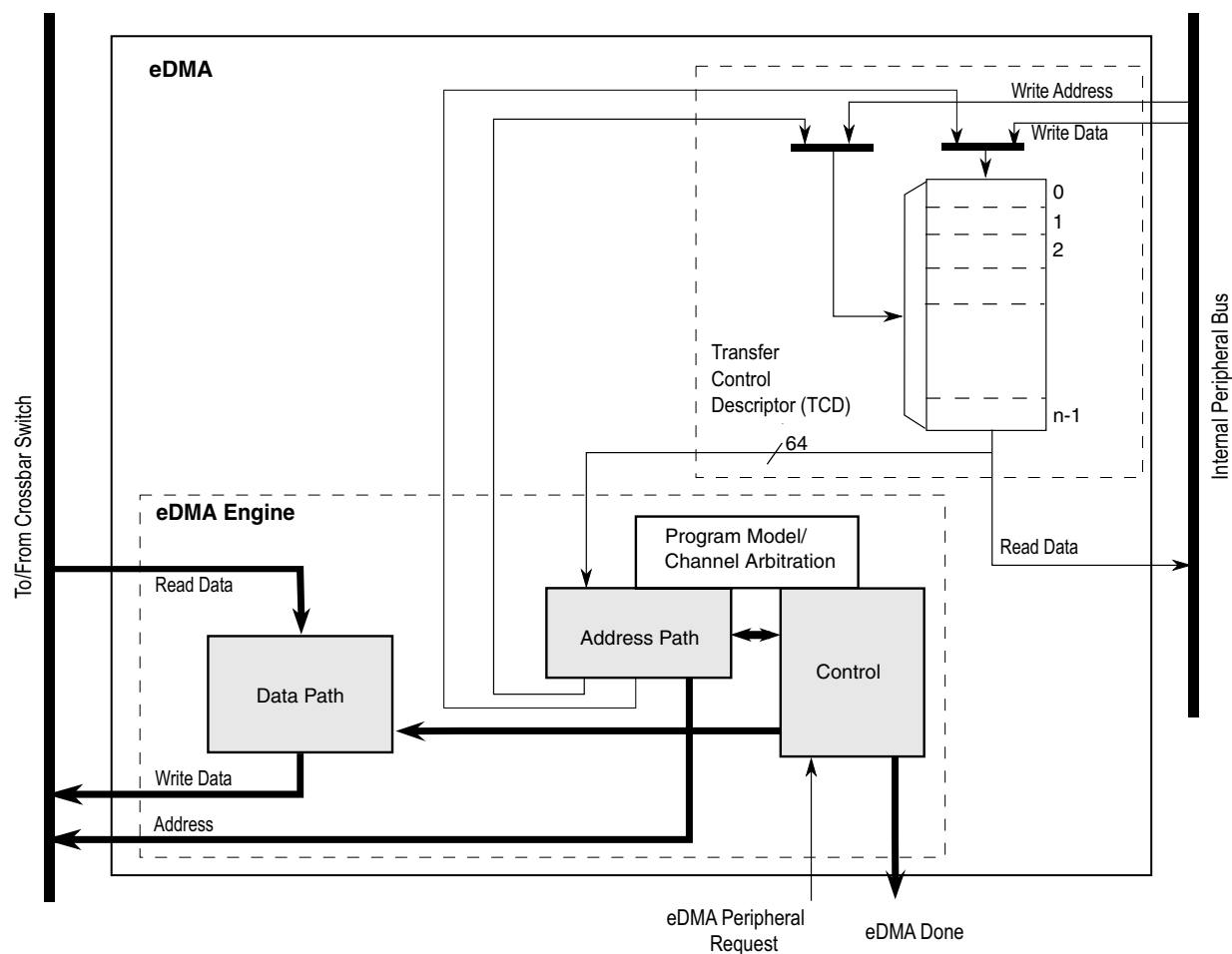


Figure 249. DMA operation, part 2

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, and CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in [Figure 250](#).

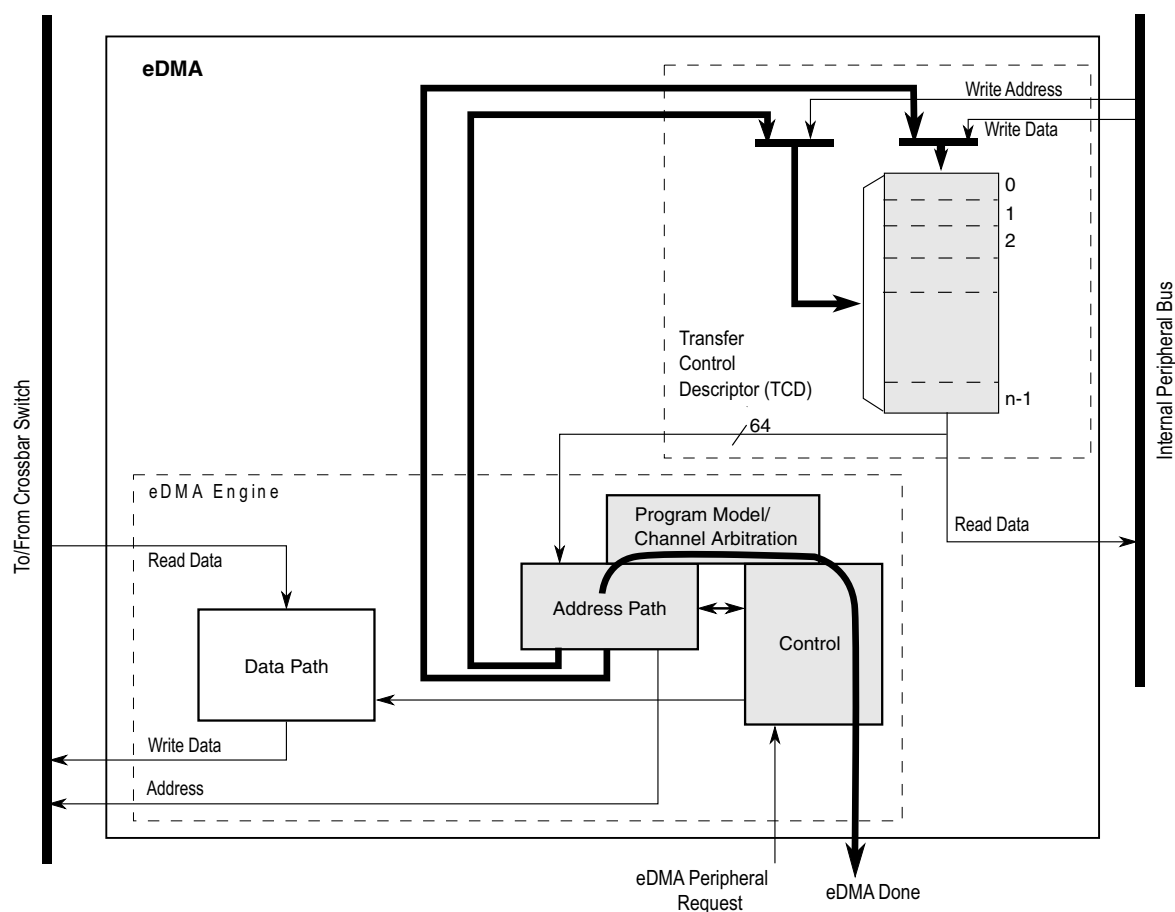


Figure 250. DMA operation, part 3

25.4.3 Error reporting and handling

Channel errors are reported in the ES register and can be caused by:

- A configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode),
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle.

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed in the list below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size, respectively.
- In Fixed Arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups.

All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when Fixed Arbitration mode is enabled.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system-bus error occurs, the channel terminates after the read or write transaction (which is already pipelined after errant access) has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the ES register is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD since the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

The occurrence of any error causes the eDMA engine to stop the active channel immediately, and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the ES register. The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

25.4.4 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer; regardless of the lower priority channel's ECP setting. This allows for a pool of low-priority, large data-moving channels to be defined. These low-priority channels can be configured so that they do not preempt each other, thus preventing a low-priority channel from consuming the preempt slot normally available to a true, high-priority channel.

25.4.5 eDMA performance

This section addresses the performance of the eDMA module, focusing on two separate metrics. In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces. In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more interesting metric. In this environment, the speed of the source and destination address spaces remains important, but the microarchitecture of the eDMA also factors significantly into the resulting metric.

The peak transfer rates for several different source and destination transfers are shown in [Table 327](#) and [Table 328](#). The following assumptions apply to those tables:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.
- All internal peripheral bus reads require one wait-state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- All internal peripheral bus accesses are 32 bits in size.
- All internal SRAM accesses are 64 bits in size.

[Table 327](#) presents a peak transfer rate comparison, measured in megabytes per second.

Table 327. eDMA peak transfer rates (Mbyte/s)

System speed, width	From internal SRAM to internal SRAM	From 32-bit internal peripheral bus to internal SRAM	From internal SRAM to 32-bit internal peripheral bus
100 MHz, 64 bits	400.0	160.0	160.0
150 MHz, 64 bits	600.0	240.0	240.0
200 MHz, 64 bits	800.0	320.0	320.0

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, it is assumed the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel. The eDMA design supports the following hardware service request sequence:

- Cycle 1: eDMA peripheral request is asserted.
- Cycle 2: The eDMA peripheral request is registered locally in the eDMA module and qualified. (TCDn_CSR[START] bit initiated requests start at this point with the registering of the user write to TCDn Word 7).
- Cycle 3: Channel arbitration begins.
- Cycle 4: Channel arbitration completes. The transfer control descriptor local memory read is initiated.
- Cycles 5–6: The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles.
- Cycle 7: The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
- Cycles 8 to x^(e): The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.

The exact timing from this point is a function of the transfer size and response times for the channel's read and write accesses. In this case of two internal peripheral bus reads and a single 64-bit internal SRAM write, the combined data phase time is five cycles.

e. "x" represents an undefined cycle value dependant on transfer sizes and response times for the channel's read and write accesses.

For a single SRAM read and two 64-bit internal peripheral bus writes, the combined data phase time is five cycles.

- Cycle $x^{(e)} + 1$: This cycle represents the data phase of the last destination write.
- Cycle $x^{(e)} + 2$: The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCDn fields into the local memory. The TCDn Word 7 is read and checked for channel linking or scatter/gather requests.
- Cycle $x^{(e)} + 3$: The appropriate fields in the first part of the TCDn are written back into the local memory.
- Cycle $x^{(e)} + 4$: The fields in the second part of the TCDn are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
- Cycle $x^{(e)} + 5$: The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every nine cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (three cycles) and SRAM-to-internal peripheral bus (three cycles), DMA requests can be processed every 10 cycles ($4 + (3 + 3)/2 + 3$). This is the time from Cycle 4 to Cycle $x + 5$. The resulting peak request rate, as a function of the system frequency, is shown in [Table 328](#). This metric represents millions of requests per second. A single read/write operation with same transfer size for the source and destination is assumed.

Table 328. eDMA peak request rate (MReq/s)

System frequency (MHz)	Request rate	
	zero wait states	with wait states
100.0	11.1	10.0
150.0	16.6	15.0
200.0	22.2	20.0

A general formula to compute the peak request rate (with overlapping requests) is:

Equation 1 $PEAKreq = freq / [entry + (1 + read_ws) + (1 + write_ws) + exit]$

where:

- PEAKreq = peak request rate
- freq = system frequency
- entry = channel startup (4 cycles)
- read_ws = wait states seen during the system bus read data phase
- write_ws = wait states seen during the system bus write data phase
- exit = channel shutdown (3 cycles)

For example: consider a system with the following characteristics:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.
- All internal peripheral bus reads require one wait state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- System operates at 150 MHz.

For an SRAM to internal peripheral bus transfer,

Equation 2 $\text{PEAKreq} = 150 \text{ MHz} / [4 + (1 + 0) + (1 + 1) + 3] \text{ cycles} = 15.0 \text{ Mreq/s}$

For an internal peripheral bus-to-SRAM transfer,

Equation 3 $\text{PEAKreq} = 150 \text{ MHz} / [4 + (1 + 1) + (1 + 0) + 3] \text{ cycles} = 15.0 \text{ Mreq/s}$

The minimum number of cycles to perform a single read/write, with zero wait states on the system bus, from a cold start (where no channel is executing and the eDMA is idle):

- 11 cycles for a software (TCDn_CSR[START] bit) request
- 12 cycles for a hardware (eDMA peripheral request signal) request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering are absorbed in or overlap the previous executing channel.

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

25.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

25.5.1 eDMA initialization

A typical initialization of the eDMA has the following sequence:

1. Write the CR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRIn registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI registers if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service by software (setting the TCDn_CSR[START] bit) or hardware (slave device asserting its eDMA peripheral request signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields (shown in [Table 329](#)) for the selected channel into its internal address path module. As the TCD is read, the first transfer is initiated on the internal bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCDn_SADDR) to the destination (as defined by

the destination address, TCDn_DADDR) continue until the specified number of bytes (TCDn_NBYTES) are transferred. When the transfer is complete, the eDMA engine's local TCDn_SADDR, TCDn_DADDR, and TCDn_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes (interrupts, major loop channel linking, and scatter/gather operations) if enabled.

Table 329. TCD control and status fields

TCDn_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

[Figure 251](#) shows how each DMA request initiates one minor-loop transfer (iteration) without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

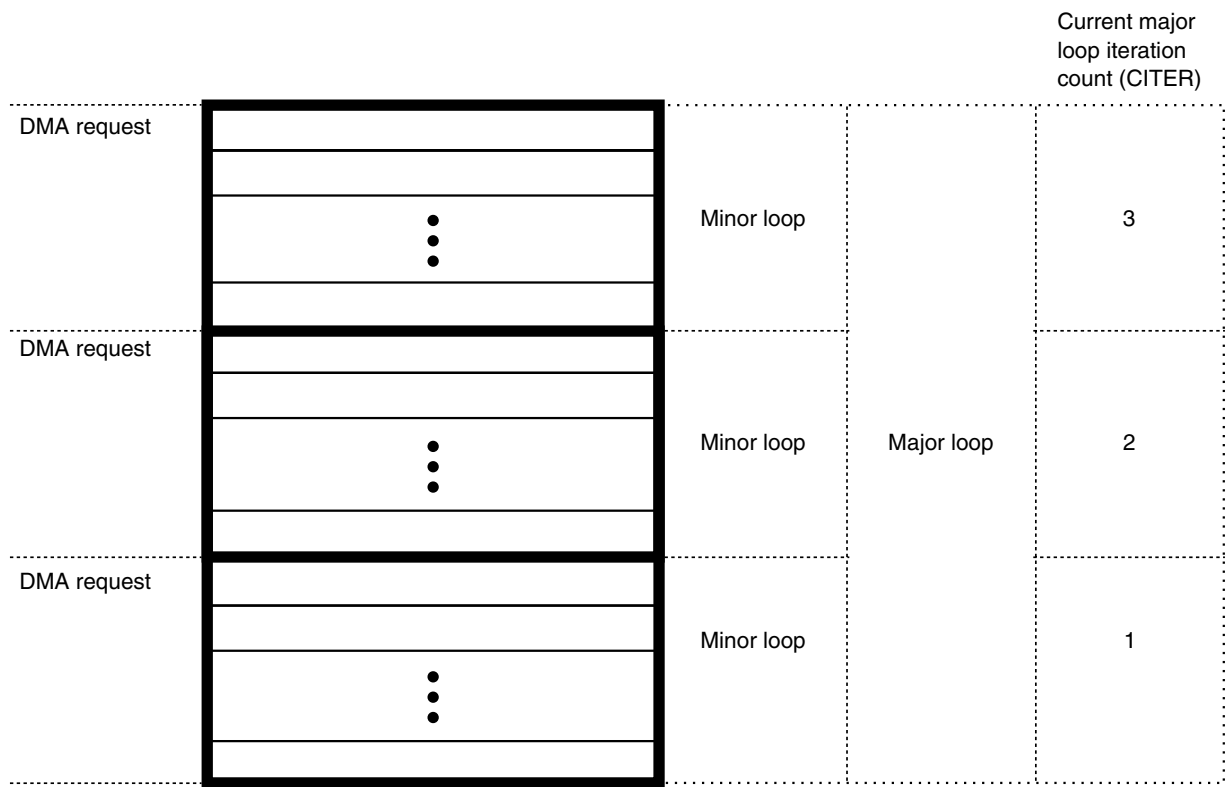


Figure 251. Example of multiple loop iterations

Figure 252 lists the memory array terms and how the TCD settings interrelate.

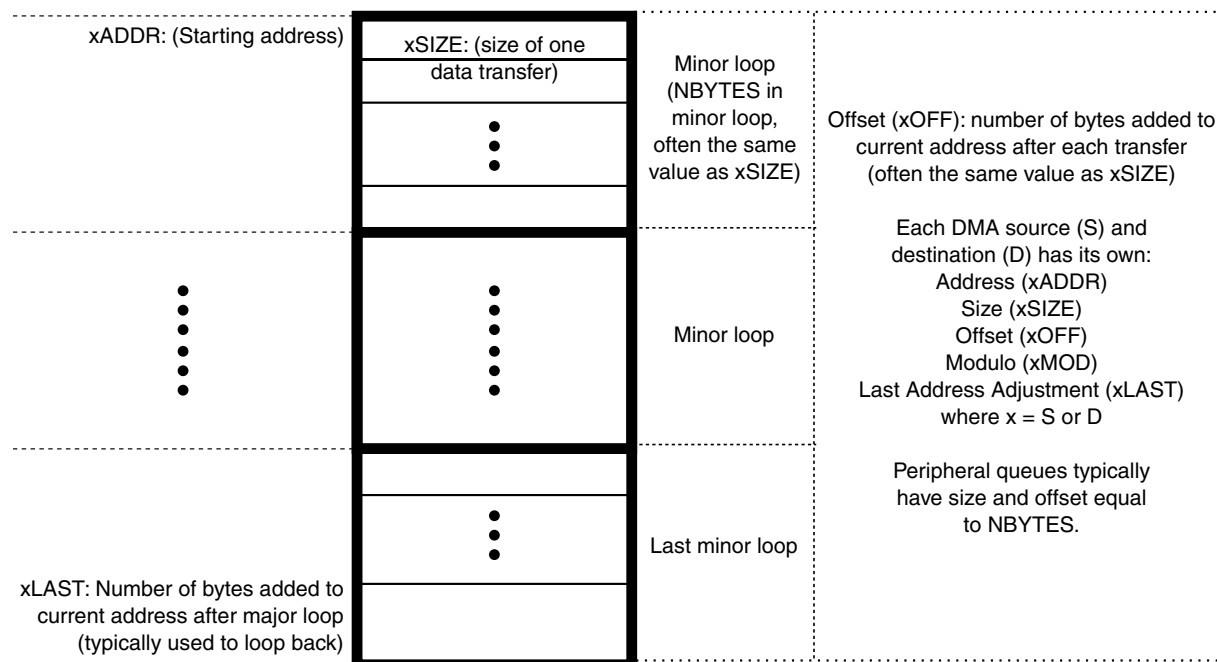


Figure 252. Memory array terms

25.5.2 DMA programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis with the exception of two errors: group priority error (ES[GPE]) and channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the ES register. If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed-group and fixed-channel arbitration modes.
2. Group 3 is the highest priority and all channels are unique in that group.
3. Group 2 is the next highest priority and has two channels with the same priority level.
4. If Group 3 has any service requests, those requests will be executed.
5. Once all of Group 3 requests have completed, Group 2 will be the next active group.
6. If Group 2 has a service request, then an undefined channel in Group 2 will be selected and a channel priority error will occur.
7. This repeats until all of the Group 2 requests have been removed or a higher priority Group 3 request comes in.

In this sequence, for item 2, the eDMA Acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the

selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest (channel/group) priority with an active request is selected, but the lowest numbered (channel/group) with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

25.5.3 DMA Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

25.5.3.1 Fixed-group arbitration, fixed-channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so the channels within one group use “fixed” priorities, and that group is assigned the highest “fixed” priority of all groups, it is possible for that group to take all the bandwidth of the eDMA controller—that is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

25.5.3.2 Round-robin group arbitration, fixed-channel arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with a service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round-robin algorithm will select the highest pending request from the next group in the round-robin sequence. Servicing continues round-robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round-robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced. The advantage of this scenario is that no one group will consume all the eDMA bandwidth. The highest priority channel selection latency is potentially greater than fixed/fixed arbitration. Excessive request rates on high priority channels could prevent the servicing of lower priority channels in the same group.

25.5.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in [Section 25.5.3.2, Round-robin group arbitration, fixed-channel arbitration](#), but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round-robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round-robin manner, any channel that generates DMA requests faster than a combination of the group round-robin service rate and the channel

service rate for its group will not prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high. All channels are treated equally. Priority levels are not used in round-robin/round-robin mode.

25.5.3.4 Fixed-group arbitration, round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in [Section 25.5.3.1, Fixed-group arbitration, fixed-channel arbitration](#) but all the channels in the highest priority group will get serviced. Service latency will be short on the highest priority group, but could potentially get very much longer and longer as the group priority decreases.

25.5.4 DMA transfer

This section discusses how to perform DMA transfers with the eDMA.

25.5.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one ($\text{TCDn_CITER} = \text{TCDn_BITER} = 1$). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the $\text{TCDn_CSR}[\text{DONE}]$ bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop, transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit-wide port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```

TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA = -16
TCDn_CSR[INT_MAJ] = 1

```

```

TCDn_CSR[START] = 1 (Should be written last after all other fields have
been
    initialized)
All other TCDn fields = 0

```

This generates the following event sequence:

1. User write to the TCDn_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCDn_CSR[DONE] = 0, TCDn_CSR[START] = 0, TCDn_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
 - a) Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
 - b) Write 32 bits to location 0x2000 → first iteration of the minor loop.
 - c) Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
 - d) Write 32 bits to location 0x2004 → second iteration of the minor loop.
 - e) Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
 - f) Write 32 bits to location 0x2008 → third iteration of the minor loop.
 - g) Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
 - h) Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: TCDn_SADDR = 0x1000, TCDn_DADDR = 0x2000, TCDn_CITER = 1 (TCDn_BITER).
7. The eDMA engine writes: TCDn_CSR[ACTIVE] = 0, TCDn_CSR[DONE] = 1, INT[n] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

25.5.4.2 Multiple requests

Besides transferring 32 bytes via two hardware requests, the next example is the same as previous. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```

TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32

```

This would generate the following sequence of events:

1. First hardware (eDMA peripheral) request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: $TCDn_CSR[DONE] = 0$, $TCDn_CSR[START] = 0$, $TCDn_CSR[ACTIVE] = 1$.
4. eDMA engine reads: channel $TCDn$ data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
 - a) Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
 - b) Write 32 bits to location 0x2000 → first iteration of the minor loop.
 - c) Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
 - d) Write 32 bits to location 0x2004 → second iteration of the minor loop.
 - e) Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
 - f) Write 32 bits to location 0x2008 → third iteration of the minor loop.
 - g) Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
 - h) Write 32 bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: $TCDn_SADDR = 0x1010$, $TCDn_DADDR = 0x2010$, $TCDn_CITER = 1$.
7. eDMA engine writes: $TCDn_CSR[ACTIVE] = 0$.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware (eDMA peripheral) requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: $TCDn_CSR[DONE] = 0$, $TCDn_CSR[START] = 0$, $TCDn_CSR[ACTIVE] = 1$.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
 - a) Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
 - b) Write 32 bits to location 0x2010 → first iteration of the minor loop.
 - c) Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
 - d) Write 32 bits to location 0x2014 → second iteration of the minor loop.
 - e) Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
 - f) Write 32 bits to location 0x2018 → third iteration of the minor loop.
 - g) Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
 - h) Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete.
14. eDMA engine writes: $TCDn_SADDR = 0x1000$, $TCDn_DADDR = 0x2000$, $TCDn_CITER = 2(TCDn_BITER)$.
15. eDMA engine writes: $TCDn_CSR[ACTIVE] = 0$, $TCDn_CSR[DONE] = 1$, $INT[n] = 1$.

16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

25.5.4.3 Modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

[Table 330](#) shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567x) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes, and the MOD field is set to 4, allowing for a 16-byte size queue.

Table 330. Modulo feature example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

25.5.5 eDMA TCD_n status monitoring

This section discusses how to monitor eDMA status.

25.5.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD_n_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD_n_CSR[START] bit and the TCD_n_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD_n_CSR[START] was set. Polling the TCD_n_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute a sequence for a software activated channel as shown in the following table.

Table 331. TCD status sequence – software activated channel

Step	TCD _n _CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing

Table 331. TCD status sequence – software activated channel (continued)

Step	TCDn_CSR bits			State
	START	ACTIVE	DONE	
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware (peripheral) initiated service requests is to read the TCDn_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute a sequence for a hardware-activated channel as shown in the following table.

Table 332. TCD status sequence – hardware activated channel

Step	TCDn_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCDn_CSR[DONE] bit. The TCDn_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

25.5.5.2 Active channel TCDn reads

The eDMA reads back the true TCDn_SADDR, TCDn_DADDR, and TCDn_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses (SADDR and DADDR) and NBYTES (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

25.5.5.3 Preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in Fixed-Group, Fixed-Channel Arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal (constantly rotating) when Round-Robin Arbitration mode is selected.

The TCDn_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCDn_CSR[ACTIVE] bits are set

simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

25.5.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD_n_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD_n_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done → set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD_n_CITER[E_LINK] = 1), the TCD_n_CITER[CITER] field uses a 9-bit vector to form the current iteration count. When minor loop linking is disabled (TCD_n_CITER[E_LINK] = 0), the TCD_n_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD_n_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note: *The TCD_n_CITER[E_LINK] bit and the TCD_n_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.*

[Table 333](#) summarizes how a DMA channel can link to another DMA channel (that is, how it can use another channel's TCD) at the end of a loop.

Table 333. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH] Link channel number when linking at end of minor loop (current iteration)	

Table 333. Channel linking parameters (continued)

Desired link behavior	TCD control field name	Description
Link at end of major loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH] Link channel number when linking at end of major loop	

25.5.7 Dynamic programming

25.5.7.1 Dynamic priority changing

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode.
2. Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing group priority levels:

1. Switch to Round-Robin Group Arbitration mode, change the group priorities, then switch back to Fixed Arbitration mode.
2. Disable all channels, change the group priorities, then enable the appropriate channels.

25.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The coherency model in [Table 334](#) is recommended when executing a dynamic channel link request.

Table 334. Coherency model for a dynamic channel link request

Step	Action
1	Write 1b to the TCD.major.e_link bit.
2	Read back the TCD.major.e_link bit.
3	Test the TCD.major.e_link request status: – If TCD.major.e_link = 1b, the dynamic link attempt was successful. – If TCD.major.e_link = 0b, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

Note: *The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.*

25.5.7.3 Dynamic scatter/gather

Dynamic scatter/gather is the process of setting the TCD.e_sg bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

Note: *The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.*

25.5.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model in [Table 335](#) may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD identification (ID).

Table 335. Coherency model for method 1

Step	Action
1	When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2	Write 1b to the TCD.d_req bit. <i>Note: Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.</i>
3	Write the TCD.dlast_sga field with the scatter/gather address.
4	Write 1b to the TCD.e_sg bit.

Table 335. Coherency model for method 1 (continued)

Step	Action
5	Read back the 16 bit TCD control/status field.
6	Test the TCD.e_sg request status and TCD.major.linkch value: – If e_sg = 1b, the dynamic link attempt was successful. – If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring). – If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

25.5.7.3.2 Method 2 (channel using major loop linking)

For a channel using major loop channel linking, the coherency model in [Table 336](#) may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 336. Coherency model for method 2

Step	Action
1	Write 1b to the TCD.d_req bit. <i>Note: Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.</i>
2	Write the TCD.dlast_sga field with the scatter/gather address.
3	Write 1b to the TCD.e_sg bit.
4	Read back the TCD.e_sg bit.
5	Test the TCD.e_sg request status: – If e_sg = 1b, the dynamic link attempt was successful. – If e_sg = 0b, read the 32 bit TCD dlast_sga field. – If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring). – If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

26 DMACHMUX

26.1 Introduction

26.1.1 Overview

The DMA Mux allows to route 64 DMA peripheral sources (called slots) to 16 DMA channels. This is illustrated in [Figure 253](#).

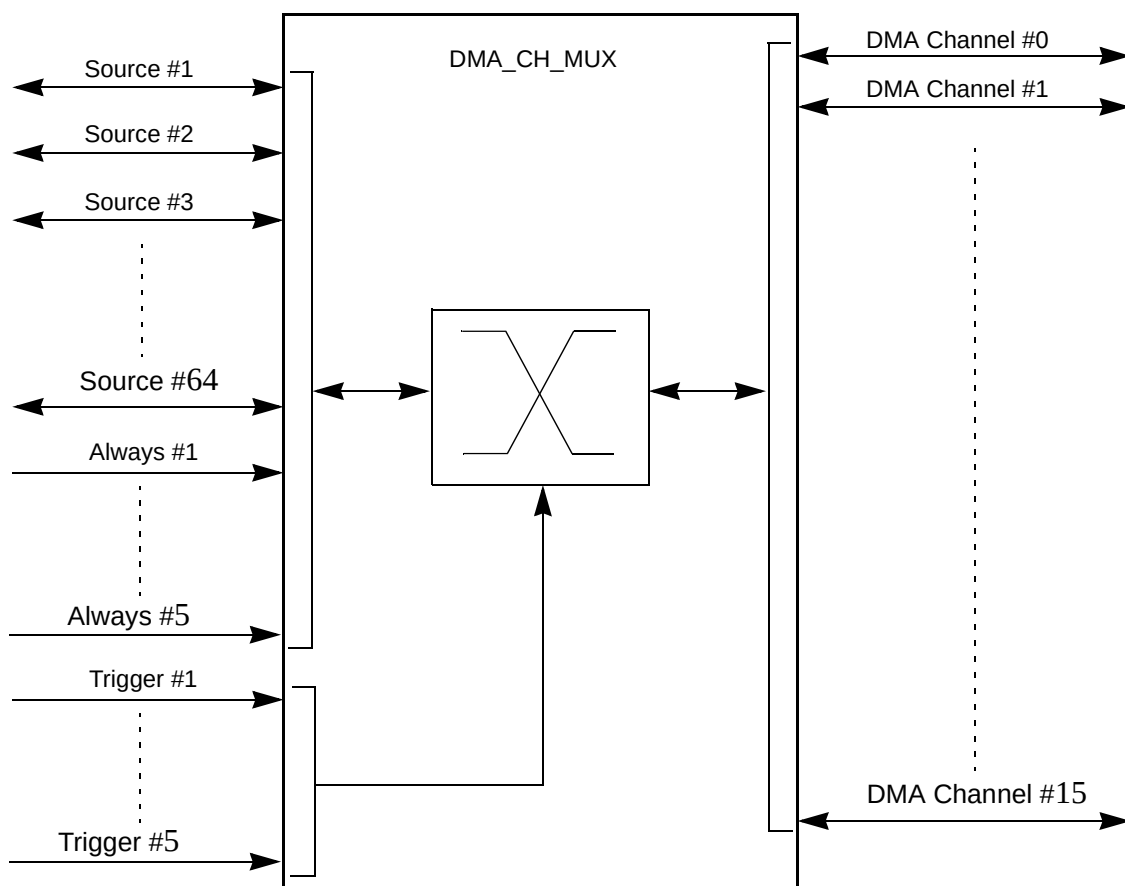


Figure 253. DMA Mux Block Diagram

26.1.2 Features

The DMA Channel Mux provides these features:

- 64 peripheral slots + 5 always-on slots can be routed to 16 channels
- 16 independently selectable DMA channels routers
 - the first 5 channels additionally provide a trigger functionality
- Each channel router can be assigned to one of 64 possible peripheral DMA slots or to one of the 5 always-on slots.

26.1.3 Modes of Operation

The following operation modes are available:

- **Disabled Mode**
In this mode, the DMA channel is disabled. Since disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA Channel Mux. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place (e.g. changing the period of a DMA trigger).
- **Normal Mode**
In this mode, a DMA source (such as DSPI transmit or DSPI receive for example) is routed directly to the specified DMA channel. The operation of the DMA Mux in this mode is completely transparent to the system.
- **Periodic Trigger Mode**
In this mode, a DMA source may only request a DMA transfer (such as when a transmit buffer becomes empty or a receive buffer becomes full) periodically. Configuration of the period is done in the registers of the Periodic Interrupt Timer (PIT). This mode is only available for channels 0-5.

26.2 External Signal Description

26.2.1 Overview

The DMA Mux has no external pins.

26.3 Memory Map and Register Definition

This section provides a detailed description of all memory-mapped registers in the DMA Mux.

[Table 337](#) shows the memory map for the DMA Mux. Note that all addresses are offsets; the absolute address may be computed by adding the specified offset to the base address of the DMA Mux.

Table 337. Module memory map

Address	Use	Access
Base + \$0x00	Channel #0 Configuration (CHCONFIG0)	R/W
Base + \$0x01	Channel #1 Configuration (CHCONFIG1)	R/W
..	..	..
Base + #n-1	Channel #n Configuration (CHCONFIG#n-1) ⁽¹⁾	R/W

1. In the table *n* refers to the *number of channels - 1*

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, CHCONFIG0 through CHCONFIG3 are accessible by a 32-bit

READ/WRITE to address 'Base + 0x00', but performing a 32-bit access to address 'Base + 0x01' is illegal.

26.3.1 Register Descriptions

The following memory-mapped registers are available in the DMA Channel Mux.

26.3.1.1 Channel Configuration Registers

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

Figure 254. Channel Configuration Registers (CHCONFIG#n)



Table 338. CHCONFIGxx Field Descriptions

Field	Description
ENBL	DMA Channel Enable. ENBL enables the DMA Channel 0 DMA channel is disabled. This mode is primarily used during configuration of the DMA Mux. The DMA has separate channel enables/disables, which should be used to disable or re-configure a DMA channel. 1 DMA channel is enabled
TRIG	DMA Channel Trigger Enable (for triggered channels only). TRIG enables the periodic trigger capability for the DMA Channel 0 Triggering is disabled. If triggering is disabled, and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel. 1 Triggering is enabled.
SOURCE	DMA Channel Source (slot). SOURCE specifies which DMA source, if any, is routed to a particular DMA channel. Please check your SoC-Guide for further details about the peripherals and their slot numbers.

Table 339. Channel and Trigger Enabling

ENBL	TRIG	Function	Mode
0	X	DMA Channel is disabled	Disabled Mode
1	0	DMA Channel is enabled with no triggering (transparent)	Normal Mode
1	1	DMA Channel is enabled with triggering	Periodic Trigger Mode

Note: Setting multiple CHCONFIG registers with the same Source value will result in unpredictable behavior.

Note: Before changing the trigger or source settings a DMA channel must be disabled via the CHCONFIG[#n].ENBL bit.

26.4 Functional description

This section provides a functional description of the DMA Mux. The primary purpose of the DMA Mux is to provide flexibility in the system's use of the available DMA channels. As such, configuration of the DMA Mux is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in [Section 26.5.2, "Enabling and Configuring Sources"](#) is followed, the configuration of the DMA MUX may be changed during the normal operation of the system.

Functionally, the DMA Mux channels may be divided into two classes: Channels, which implement the normal routing functionality plus periodic triggering capability, and channels, which implement only the normal routing functionality.

26.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 5 channels of the DMA Mux provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames or packets at fixed intervals without the need for processor intervention. The trigger is generated by the Periodic Interrupt Timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. Please refer to the Periodic Interrupt Timer chapter for more information on this topic.

Note: Because of the dynamic nature of the system (i.e. DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

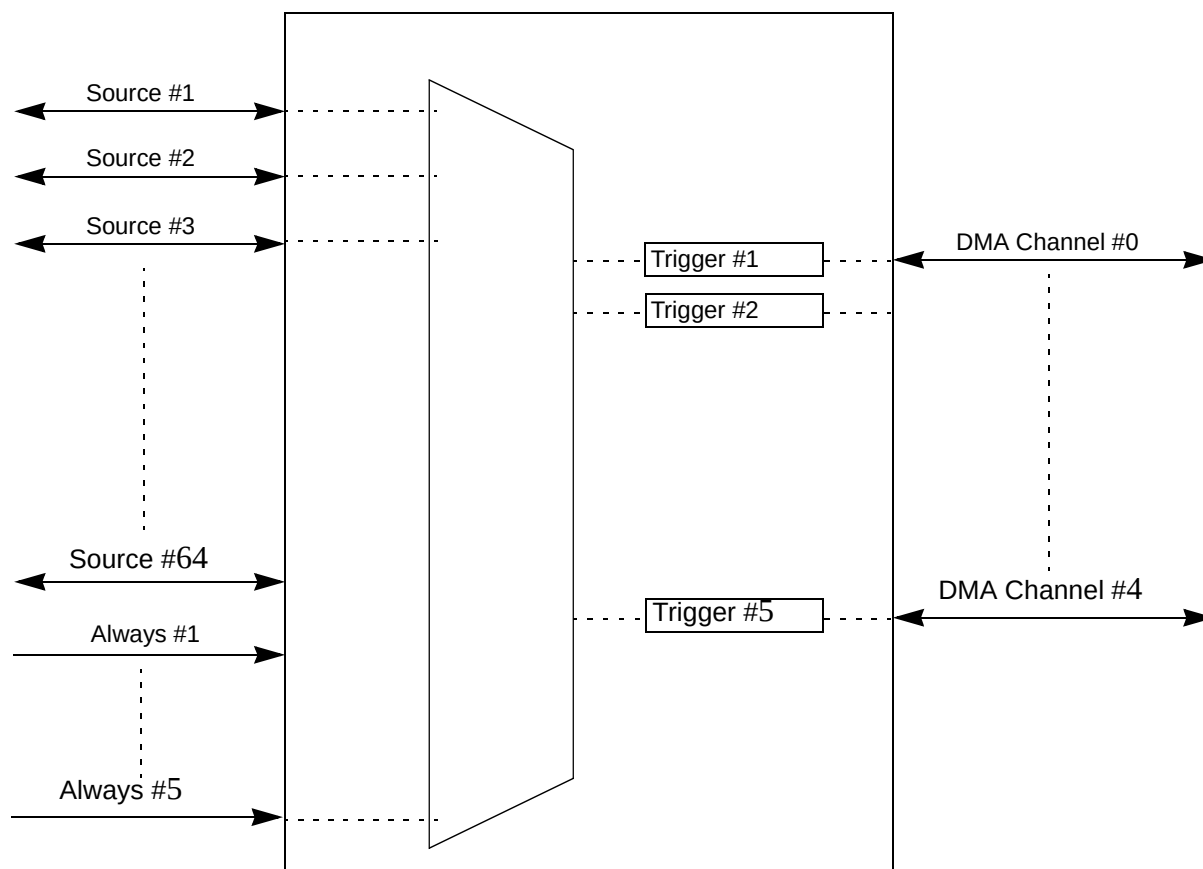


Figure 255. DMA Mux triggered channels

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the Peripheral to the DMA until a trigger event has been seen. This is illustrated in [Figure 256](#).

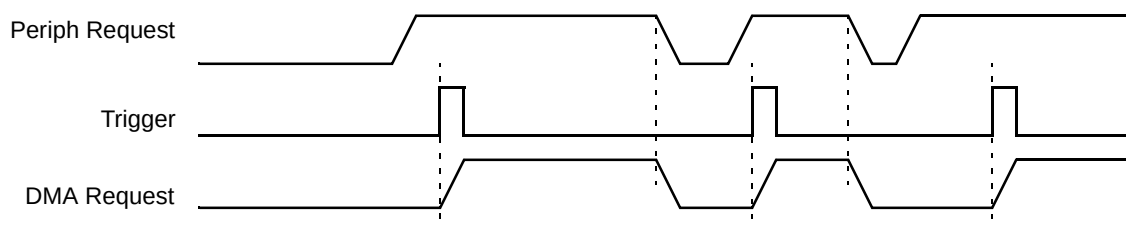


Figure 256. DMA Mux Channel Triggering: Normal Operation

Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request AND the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, that trigger will be ignored. This situation is illustrated in [Figure 257](#).

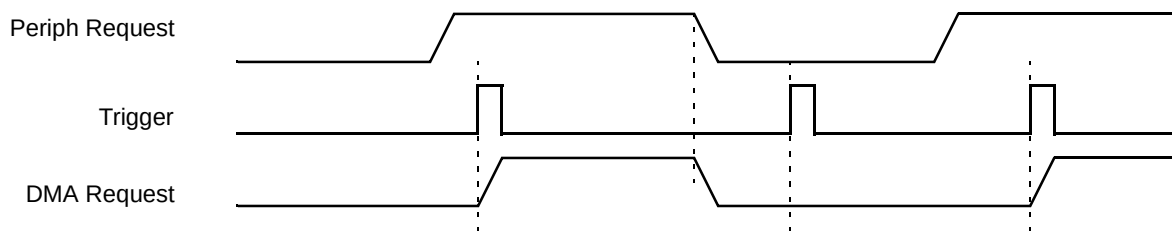


Figure 257. DMA Mux Channel Triggering: Ignored Trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus. As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. Once setup, the SPI will request DMA transfers (presumably from memory) as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5μs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO Ports to drive or sample waveforms. By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger (i.e.-resolution, range of values, etc.) may be found in the Periodic Interrupt Timer (PIT) chapter.

26.4.2 DMA Channels with no triggering capability

The other channels of the DMA Mux provide the normal routing functionality as described in [Section 26.1.3, “Modes of Operation.”](#)

26.4.3 “Always Enabled” DMA Sources

In addition to the peripherals that can be used as DMA sources, there are 5 additional DMA sources that are “always enabled”. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the “always enabled” sources provide no such “throttling” of the data transfers. These sources are most useful in the following cases:

- Doing DMA transfers to/from GPIO - Moving data from/to one or more GPIO pins, either un-throttled (i.e.-as fast as possible), or periodically (using the DMA triggering capability).
- Doing DMA transfers from memory to memory - Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Doing DMA transfers from memory to the external bus (or vice-versa) - Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation - Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, a “always enabled” DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require a new “start” event be sent. This can either be a new software activation, or a transfer request from the DMA Channel Mux. The options for doing this are:

- Transfer all data in a single minor loop. By configuring the DMA to transfer all of the data in a single minor loop (i.e.-major loop counter = 1), no re-activation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will incur on the system. For this option, the DMA channel should be disabled in the DMA Channel Mux.
- Use explicit software re-activation. In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to re-activate the channel (by writing to the DMA registers) *after every minor loop*. For this option, the DMA channel should be disabled in the DMA Channel Mux.
- Use a “always enabled” DMA source. In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA Channel Mux does the channel re-activation. For this option, the DMA channel should be enabled and pointing to an “always enabled” source. Note that the re-activation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

26.5 Initialization/Application Information

26.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See [Section 26.3.1, “Register Descriptions”](#)). In summary, after reset, all channels are disabled and must be explicitly enabled before use.

26.5.2 Enabling and Configuring Sources

Enabling a source with periodic triggering

1. Determine with which DMA channel the source will be associated. Note that only the first 5 DMA channels have periodic triggering capability.
2. Clear the ENBL and TRIG bits of the DMA channel
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point
4. Configure the corresponding timer
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL and TRIG bits are set

Example 1 Configure source #5 Transmit for use with DMA Channel 2, with periodic triggering capability

1. Write 0x00 to CHCONFIG2 (Base Address + 0x02)
2. Configure Channel 2 in the DMA, including enabling the channel
3. Configure a timer for the desired trigger interval
4. Write 0xC5 to CHCONFIG2 (Base Address + 0x02)

The following code example illustrates steps #1 and #4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR      0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

```
In File main.c:
#include "registers.h"

:
:
*CHCONFIG2 = 0x00;
*CHCONFIG2 = 0xC5;
```

Enabling a source without periodic triggering

1. Determine with which DMA channel the source will be associated. Note that only the first 5 DMA channels have periodic triggering capability.
2. Clear the ENBL and TRIG bits of the DMA channel
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL is set and the TRIG bit is cleared

Example 2 Configure source #5 Transmit for use with DMA Channel 2, with no periodic triggering capability.

1. Write 0x00 to CHCONFIG2 (Base Address + 0x02)
2. Configure Channel 2 in the DMA, including enabling the channel
3. Write 0x85 to CHCONFIG2 (Base Address + 0x02)

The following code example illustrates steps #1 and #3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR      0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
```

```
volatile unsigned char *CHCONFIG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c**:

```
#include "registers.h"

:

:

*CHCONFIG2 = 0x00;
*CHCONFIG2 = 0x85;
```

Disabling a source

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCONFIG registers. Additionally, some module specific configuration may be necessary. Please refer to the appropriate section for more details.

Switching the source of a DMA Channel

1. Disable the DMA channel in the DMA and re-configure the channel for the new source
2. Clear the ENBL and TRIG bits of the DMA channel
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL and TRIG bits are set

Example 3 Switch DMA Channel 8 from source #5 transmit to source #7 transmit

1. In the DMA configuration registers, disable DMA channel 8 and re-configure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCONFIG8 (Base Address + 0x08)
3. Write 0x87 to CHCONFIG8 (Base Address + 0x08). (In this example, setting the TRIG bit would have no effect, due to the assumption that channels 8 does not support the periodic triggering functionality).

The following code example illustrates steps #2 and #3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR      0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
```

```
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c**:

```
#include "registers.h"
:
:
*CHCONFIG8 = 0x00;
*CHCONFIG8 = 0x87;
```

27 Clocking

27.1 Introduction

This chapter describes the architecture for the system level clocks and includes the following information:

- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers

Note: *.This chapter covers only clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.*

The system clock (CLKOUT) for the computational shell (which includes the three main CPUs (Main Core_1, Main Core_0, Checker Core_0s) as well as all modules that run in the high speed domain), memories, and debug logic is independent from the peripheral clocks. This allows CLKOUT to be a frequency modulated clock to reduce electromagnetic emissions, while having a precise clock for the peripheral timer and communication functions. It also allows the system clock to be reduced during low computational activity periods to help reduce power consumption, while providing a fixed frequency to the communication interfaces and timers.

This device boots from the internal 16 MHz RC oscillator (IRCOSC), and uses this as a backup clock in the event of a PLL or oscillator failure (if backup clock is enabled). The backup clock can be enabled in the Mode Entry Module (MC_ME). See the “Mode Entry Module (MC_ME)” chapter for details.

There are three possible ways to provide the source clock:

- External oscillator
- External crystal
- Internal 16 MHz RC oscillator

From one of these input sources, the internal clocks are generated from one of the two PLLs, PLL0 and PLL1, using PLL0_PHI and PLL1_PHI outputs, respectively. These two clocks, along with the XOSC and IRCOSC, can be selected to drive system peripherals depending on the configuration of the Auxiliary Clock Selectors in the \ Module (MC_CGM). The PHI1 output of PLL0 can also be used as the clock source for PLL1. Eleven Auxiliary Clock Selectors are utilized, allowing developers to select an independent clock source for each of the system peripherals. There is one additional clock selector that is used exclusively for the system clock.

Each of the outputs of the module clock selectors have individual dividers that allow for more clock frequency granularity with the ability to divide the clock selector outputs by up to 512 for a given peripheral. Clock Monitor Units (CMUs) are connected to outputs of clock dividers as shown in [Figure 258 \(Clock generation \(part 1 of 2\)\)](#) to verify that their clock frequencies stay within necessary operating limits. The action taken when a CMU detects an issue with the monitored clock depends on the CMU configuration. When an issue with a clock is found, the CMUs can signal an interrupt or initiate a system reset.

27.2 Clock generation

The top-level clock generation architecture can be seen in [Figure 258](#). See the Clock Generation Module (MC_CGM) chapter for details on the Fractional Clock Dividers. All clock selectors and dividers are located and programmed in the MC_CGM, with the exception of the Hardware Security Module (HSM) clock divider. All dividers are integer dividers (1, 2, 3, ..., n) with the ranges shown in the divider blocks in [Figure 258](#). All dividers connected to the System Clock Selector must have 50% duty cycle outputs.

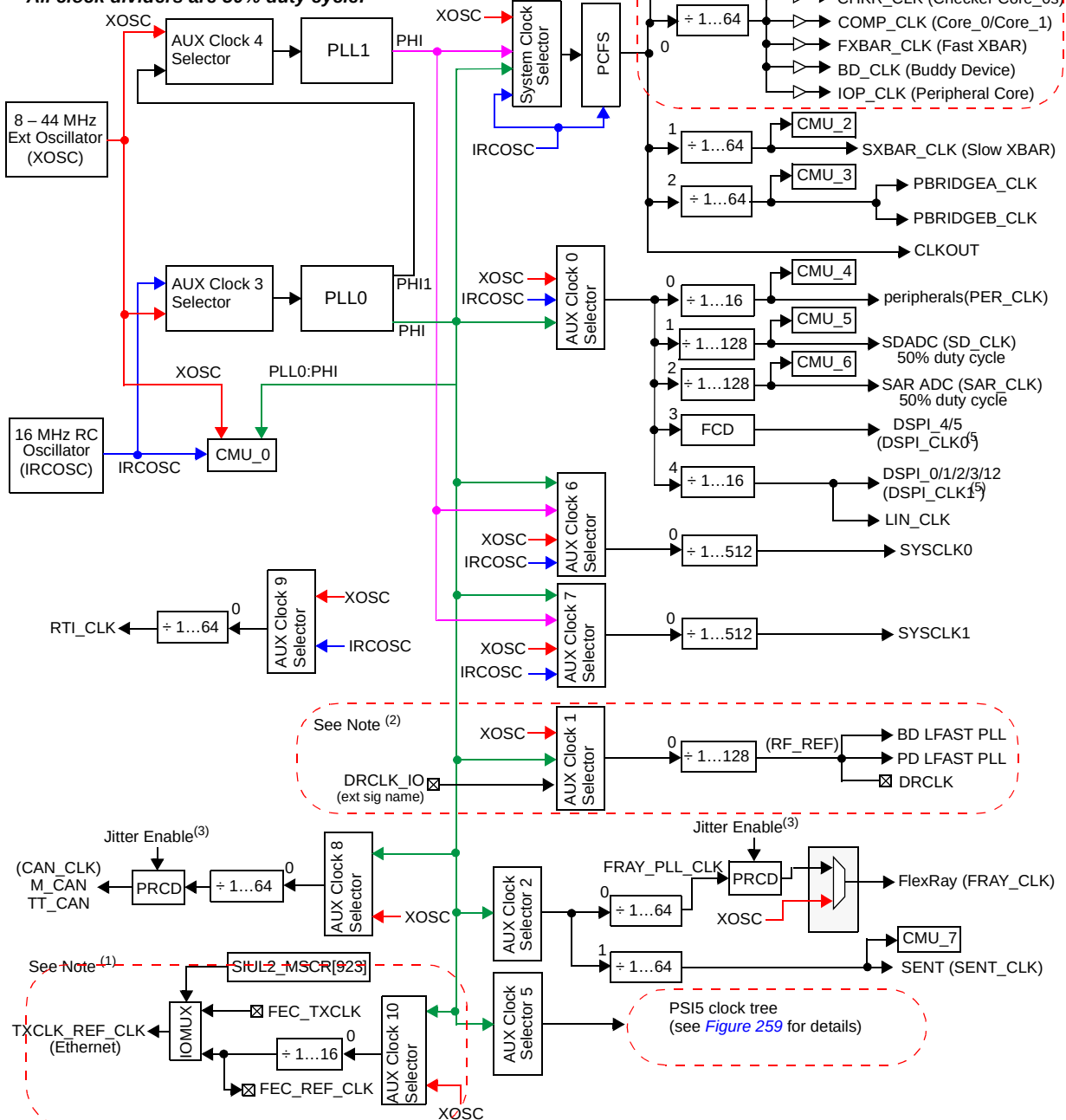
Note: CMU1...CMU10 in [Figure 258](#) use IRCOSC for their reference clock. However, the user can select either IRCOSC or XOSC for the reference in CMU0.

Figure 258. Clock generation (part 1 of 2)

Note: All dividers shown in the diagram are integer dividers with a range of 1, 2, 3,..., n.

FCD are fractional clock dividers.

All clock dividers are 50% duty cycle.



1. See [Figure 263](#) for Ethernet clocking details
2. See [Figure 262](#) for LFAST clocking details
3. See the "Password and Device Security Module (PASS)" chapter for Jitter Enable details.
4. The COMP_CLK and CHKR_CLK clock trees are isolated downstream of System Clock Divider 0. Isolation is maintained in the physical design of the clock trees.
5. DSPI_CLK1 and DSPI_CLK0 are the protocol clocks as described in the "Deserial Serial Peripheral Interface (DSPI)" chapter.
6. See the MC_CGM chapter for details of the Fractional Clock Divider (FCD).

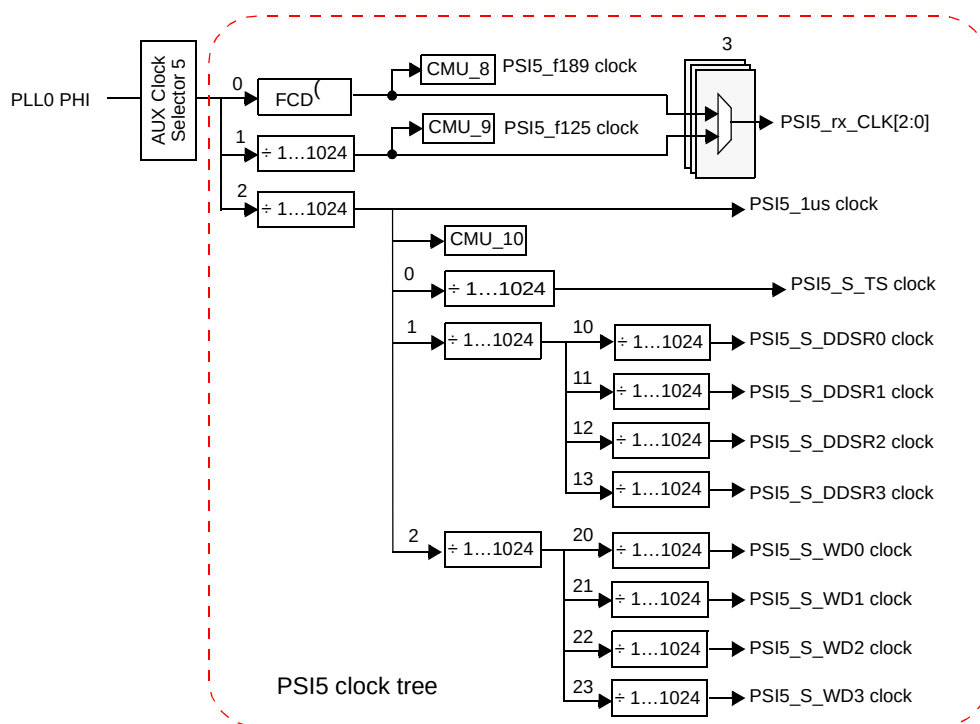


Figure 259. Clock generation (part 2 of 2)

27.2.1 MC_CGM registers

Table 340 shows the relationship between the clocks in Figure 258 and the MC_CGM registers.

Table 340. MC_CGM relationship to clocks⁽¹⁾

MC_CGM register(s)	Register description	Output clock	CMU ⁽²⁾
CGM_SC_DC0	System Clock Divider Configuration 0	Checker Core_0s (CHKR_CLK) Main Core_0 (COMP_CLK) Main Core_1 (COMP_CLK) Peripheral Core_2 (IOP_CLK) Fast XBAR (FXBAR_CLK) Buddy Device (BD_CLK)	CMU_1
CGM_SC_DC1	System Clock Divider Configuration 1	Slow XBAR (SXBAR_CLK)	CMU_2
CGM_SC_DC2	System Clock Divider Configuration 2	PBRIDGEA_CLK PBRIDGEB_CLK	CMU_3
CGM_AC0_DC0 CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	Peripherals (PER_CLK)	CMU_4
CGM_AC0_DC1 CGM_AC0_SC	Aux Clock 0 Divider Configuration 1 Aux Clock 0 Select Control	SDADC (SD_CLK)	CMU_5
CGM_AC0_DC2 CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	SARADC (SAR_CLK)	CMU_6

Table 340. MC_CGM relationship to clocks⁽¹⁾

MC_CGM register(s)	Register description	Output clock	CMU ⁽²⁾
CGM_AC0_DC3 CGM_AC0_SC	Aux Clock 0 Divider Configuration 3 Aux Clock 0 Select Control	DSPI_4/5 (DSPI_CLK0)	—
CGM_AC0_DC4 CGM_AC0_SC	Aux Clock 0 Divider Configuration 4 Aux Clock 0 Select Control	DSPI_0/1/2/3/12 (DSPI_CLK1) LINFlexD (LIN_CLK)	—
CGM_AC1_DC0 CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	BD LFAST PLL PD LFAST PLL DRCLK	—
CGM_AC2_DC0	Aux Clock 2 Divider Configuration 0	FRAY_PLL_CLK	—
CGM_AC2_DC1	Aux Clock 2 Divider Configuration 1	SENT (SENT_CLK)	CMU_7
CGM_AC3_SC	Aux Clock 3 Select Control	PLL0	—
CGM_AC4_SC	Aux Clock 4 Select Control	PLL1	—
CGM_AC5_DC0	Aux Clock 5 Divider Configuration 0	PSI5_f189_CLK	CMU_8
CGM_AC5_DC1	Aux Clock 5 Divider Configuration 1	PSI5_f125_CLK	CMU_9
CGM_AC5_DC2	Aux Clock 5 Divider Configuration 2	PSI5_1μs_CLK	CMU_10
CGM_AC5_CDC0	Aux Clock 5 Cascaded Divider Configuration 0	PSI5_S_TS_CLK	—
CGM_AC5_CDC1	Aux Clock 5 Cascaded Divider Configuration 1	PSI5_S_DDSR_CLK	—
CGM_AC5_CDC2	Aux Clock 5 Cascaded Divider Configuration 2	PSI5_S_WD_CLK	—
CGM_AC5_CDC10	Aux Clock 5 Cascaded Divider Configuration 10	PSI5_S_DDSR0_CLK	—
CGM_AC5_CDC11	Aux Clock 5 Cascaded Divider Configuration 11	PSI5_S_DDSR1_CLK	—
CGM_AC5_CDC12	Aux Clock 5 Cascaded Divider Configuration 12	PSI5_S_DDSR2_CLK	—
CGM_AC5_CDC13	Aux Clock 5 Cascaded Divider Configuration 13	PSI5_S_DDSR3_CLK	—
CGM_AC5_CDC20	Aux Clock 5 Cascaded Divider Configuration 20	PSI5_S_WD0_CLK	—
CGM_AC5_CDC21	Aux Clock 5 Cascaded Divider Configuration 21	PSI5_S_WD1_CLK	—
CGM_AC5_CDC22	Aux Clock 5 Cascaded Divider Configuration 22	PSI5_S_WD2_CLK	—
CGM_AC5_CDC23	Aux Clock 5 Cascaded Divider Configuration 23	PSI5_S_WD3_CLK	—
CGM_AC6_DC CGM_AC6_SC	Aux Clock 6 Divider Configuration 1 Aux Clock 6 Select Control	SYSCLK0	—
CGM_AC7_DC0 CGM_AC7_SC	Aux Clock 7 Divider Configuration 0 Aux Clock 7 Select Control	SYSCLK1	—

Table 340. MC_CGM relationship to clocks⁽¹⁾

MC_CGM register(s)	Register description	Output clock	CMU ⁽²⁾
CGM_AC8_DC0 CGM_AC8_SC	Aux Clock 8 Divider Configuration 0 Aux Clock 8 Select Control	M_CAN, TT_CAN (CAN_CLK)	—
CGM_AC9_DC0 CGM_AC9_SC	Aux Clock 9 Divider Configuration 0 Aux Clock 9 Select Control	RTI_CLK	—
CGM_AC10_DC0 CGM_AC10_SC	Aux Clock 10 Divider Configuration 0 Aux Clock 10 Select Control	Ethernet (FEC_REF_CLK)	—

1. See the Clock Generation Module (MC_CGM) for specifics on register implementation.

2. CMU_0 not shown in table since it does not monitor an output clock from the MC_CGM.

27.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in [Table 342](#).

Note: The output of the system clock selector mux has a maximum frequency of 200 MHz

Table 341. Maximum system level clock frequencies^{(1),(2)}

Clock Selector	Nominal programmed Max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
System clock selector ⁽³⁾	200	0	CMU_1 IOP core FXBAR Core_0 Core_0s Core_1	200
		1	CMU_2 SXBAR PBRIDGE - GTM/DMA/CAN ⁽⁴⁾	100
		2	CMU_3 PBRIDGEA_CLK PBRIDGEB_CLK	50
		3	Computational Cores (Core_0, Core_0s, Core_1)	—
		—	HSM_CLK (Peripheral Clock)	100
		—	BD_CLK (Buddy Device)	200

Table 341. Maximum system level clock frequencies^{(1),(2)}

Clock Selector	Nominal programmed Max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 0 Selector	400	0	PER_CLK	80
		1	SD_CLK (SD ADC)	16
		2	SAR_CLK (SAR ADC)	16
		3	DSPI_CLK0 (DSPI_4/5/6)	100
		4	DSPI_CLK1 (DSPI_0/1/2/3/12) LIN_CLK	100
Aux Clock 1 Selector	400	0	BD LFAST PLL PD LFAST PLL	26
Aux Clock 2 Selector	400	0	FRAY_CLK (FlexRay)	40
		1	SENT_CLK	100
Aux Clock 3 Selector	See Data Sheet		PLL0	—
Aux Clock 4 Selector	See Data Sheet		PLL1	—
Aux Clock 5 Selector	400	0	PSI5_f189_CLK (FCD output)	6.048
		1	PSI5_f125_CLK	4
		2	PSI5_1us_CLK	1
		CDC0	PSI5_S_TS_CLK	25
		CDC1	PSI5_S_DDSR_CLK	25
		CDC10	PSI5_S_DDSR0_CLK	25
		CDC11	PSI5_S_DDSR1_CLK	25
		CDC12	PSI5_S_DDSR2_CLK	25
		CDC13	PSI5_S_DDSR3_CLK	25
		CDC2	PSI5_S_WD_CLK	25
		CDC20	PSI5_S_WD_CLK0	25
		CDC21	PSI5_S_WD_CLK1	25
		CDC22	PSI5_S_WD_CLK2	25
		CDC23	PSI5_S_WD_CLK3	25
Aux Clock 6 Selector	400	0	SYSClk0	100
Aux Clock 7 Selector	400	0	SYSClk1	100
Aux Clock 8 Selector	400	0	CAN_CLK	50

Table 341. Maximum system level clock frequencies^{(1),(2)}

Clock Selector	Nominal programmed Max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 9 Selector	40	0	RTI_CLK	40
Aux Clock 10 Selector	400	0	FEC_REF_CLK	50
—	—	—	FEC_TXCLK	25
—	—	—	FEC_RXCLK	25

1. The user is responsible to verify that these values are not exceeded.
2. See [Figure 258](#) for clocking architecture overview.
3. System clock selection is a component of the device run mode switching process. See the Mode Entry Module (ME_ME) chapter for details.
4. Each peripheral bridge (PBRIDGE_A, PBRIDGE_B) is capable of clocking the bus interface of each peripheral slot individually at the slow XBAR frequency, or at the slow XBAR frequency divided by two. The GTM is clocked at the slow XBAR frequency, and all other peripherals on both AIPS bridges are clocked at the slow XBAR frequency divided by two.

In order to maintain synchronization between the different system clock branches (for example, SYS_CLK output from the System Clock Selector), there are limitations on the frequencies of those clocks. The system clocks must be binary multiples of each other with the frequency relationships described in the electrical specifications.

27.3.1 JTAG frequencies

[Table 343](#) shows the maximum frequencies of the JTAG interface.

Table 343. JTAG frequencies

Device	Configuration	e200z4 max. internal JTAG frequency	Buddy device max. RWA internal frequency	Max. external JTAG frequency	Notes
PD ⁽¹⁾	Unconfigured clock	—	N/A		Assumes 16 MHz clock on PD (IRC) with +/- 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
	Clock configured to 200 MHz	100 MHz			JTAG usable at 1/2 CPU frequency
ED ⁽²⁾	BD only powered	N/A	22.4 MHz	See electrical characteristics for maximum usable external JTAG frequency	BD clocked by internal 64 MHz RC clock (RCOSC64) with +/- 30% tolerance and BD RWA usable at 1/2 frequency
	PD and BD powered, unconfigured clock	—	—		ED clocked by internal 8 MHz RC clock (IRC) with +/- 1.5% tolerance, and BD RWA usable at 1/2 frequency
	PD and BD powered, PD clock configured clock to 200 MHz	100 MHz	100 MHz		100 MHz JTAG achievable using LFAST mode debug, or s/w mode JTAGM. JTAG messages are generated internally by JTAGM and stay within BD and PD domain. The external tool interface is LFAST only

1. PD = Production Die

2. ED = Emulation and Debug Device = Production Die + Buddy Die

27.4 Default clock configuration

At power up, the IRCOSC is the default clock for the entire system. All system clock dividers are set to '1' at power on reset, except for the HSM, which is set to its maximum value.

27.5 Clock sources

The following clock sources are described in this section:

- PLL0
- PLL1 (FMPLL)
- External oscillator (XOSC)
- External clock (EXTAL bypass)
- 16 MHz internal RC oscillator (IRCOSC)

27.5.1 PLL

The PLL is a dual-PLL providing separate system and peripheral clocks as shown in [Figure 258](#). The PLLs are disabled after power is cycled and must be enabled by software. The block diagram for the Dual PLL is shown in [Figure 260](#).

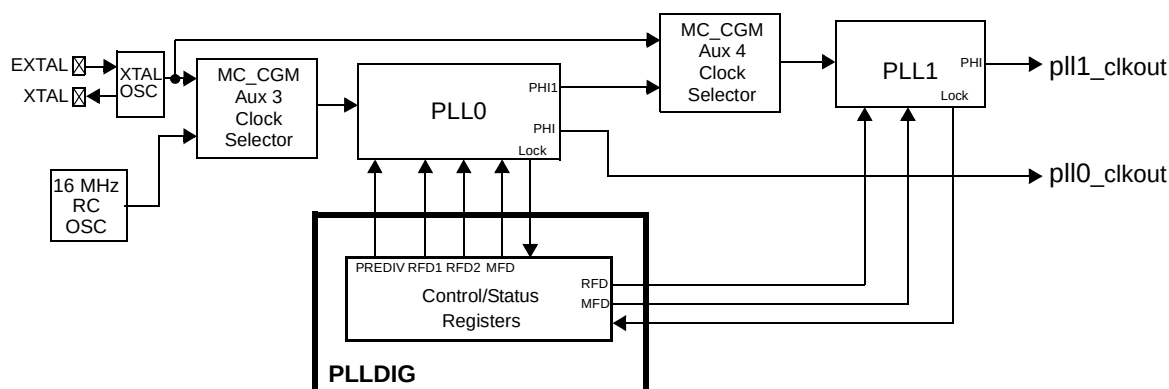


Figure 260. SPC57EM80xx Dual PLL digital interface block diagram

27.5.1.1 PLL0 — base PLL (non-FM)

PLL0 is the primary PLL. This PLL provides a non-Frequency Modulated clock to the peripheral IPs and the reference clock to PLL1.

27.5.1.1.1 PLL0 input clocks

The input clock sources for PLL0 are the XOSC, IRCOSC, and external reference clock. The external reference clock is input on the EXTAL pin.

27.5.1.1.2 PLL0 output clocks

The output clocks from PLL0 are PHI and PHI1. The PHI output clock drives the peripheral clock and the system clock when PLL1 is not locked or active. The PHI1 output clock provides the input reference clock for PLL1.

27.5.1.2 PLL1 — FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is used to drive the system clock.

27.5.1.2.1 PLL1 input clocks

The input clocks to PLL1 are the PHI1 output clock from PLL0 and the external reference clock.

27.5.1.2.2 PLL1 output clocks

The output clock from PLL1 is the PHI clock, which drives the system clock. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

27.5.1.2.3 PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of the PLLs. Details are provided in the “Dual PLL Digital Interface” chapter.

27.5.1.2.4 PLL register write protection

The PLL registers with write protection are defined in [Table 344](#). The protected registers have write protection with both soft and hard locking. A soft lock can be unlocked by software after being previously locked. A hard lock is only unlocked by a reset once locked. The REG_PROT module is used to implement the PLL register write protection.

Table 344. PLL register write protection⁽¹⁾

Offset	Register	Protection
00h	PLL0 Control Register (PLL0CR)	Yes
04h	PLL0 Status Register (PLL0SR)	No
08h	PLL0 Divider Register (PLL0DIVR)	Yes
20h	PLL1 Control Register (PLL1CR)	Yes
24h	PLL1 Status Register (PLL1SR)	No
28h	PLL1 Divider Register (PLL1DIVR)	Yes
2Ch	PLL1 Frequency Modulation Register (PLL1FMR)	Yes
30h	PLL1 Fractional Divider Register (PLL1FDR)	Yes

1. See Register Protection Configuration chapter for bit field details.

27.5.1.3 PLL register reset values

All reset values for the PLLDIG registers that are device specific are shown in [Table 345](#). All other register reset values are shown in the Dual PLL Digital Interface chapter.

Table 345. Dual PLL Digital Interface register reset values

Offset (hex)	Register	Reset value (hex)
0008	PLL0DV — PLL0 Divider Register	0000_0000
0028	PLL1DV — PLL1 Divider Register	0000_0000

27.5.1.4 Loss of clock detection

Loss of clock detection should be enabled at all times when the PLLs are enabled. There is a software option to generate a reset or interrupt in the event of a loss of clock condition. An optional backup clock is implemented within the PLL that software may enable or disable.

27.5.2 External oscillator (XOSC)

The external oscillator (XOSC) is provided as a reference for the on-chip PLLs. It can also be used as a clock source to the ADCs, timer, serial interfaces, RTI, LFAST, and as a system clock source. It is available for observation on either of the CLKOUT pins, and is used as a reference to calibrate the IRCOSC frequency.

The external oscillator allows a crystal or external clock to be used as the reference clock for the microcontroller. The XOSC has the following features:

- Internal load capacitors for 20 MHz/40 MHz crystal oscillators
- Automatic Loop Control (ALC) to remove need for external series resistor
- Gain selection for ≤ 20 MHz and 40 MHz crystals to allow for optimal startup margin without overdrive issues
- Reference clock to PLL0
- Reference clock to CMU0 (IRCOSC trimming)
- Option to drive the CAN and FlexRay protocol clocks directly from the XOSC

The XOSC provides support for 8 – 44 MHz crystal inputs, has integrated load capacitors and ALC to remove the need for external series resistor.

27.5.2.1 XOSC Start-up

Enabling of the oscillator at startup is determined by a bit in the UTEST row of flash memory (UTEST DCF Miscellaneous row, bit 7). The default value has the XOSC disabled (UTEST Misc bit 7 = 0 (XOSC EN)).

If enabled, the oscillator is started in phase 3 of the reset sequence, after negation of the internal POR circuits, but while the external PORST input is asserted. This is done to allow fast application start-up time while accounting for settling time of the power regulators.

The selection of internal or external load capacitors on the XTAL/EXTAL pins is also determined by a bit in the UTEST row of flash memory. The default value is external load capacitors (UTEST Misc bit 6 = 1 (XOSC EXT LOAD)). Load capacitor values are determined by the crystal manufacturer data sheet requirements, while accounting for stray PCB and on-chip capacitance. See the device data sheet for on-chip capacitance values.

When using the internal load capacitors for the oscillator, the startup value of the capacitors is stored in the UTEST row of the flash memory. During phase 3 of the device reset sequence, the oscillator UTEST values are read and driven to the oscillator. The enable/disable state of the oscillator is captured in the MC_ME module at this time. After reset, the oscillator can be enabled/disabled by software in the MC_ME module. Internal load capacitor selection is maintained by the oscillator if disabled after reset. The default reset value for the XOSC trimming capacitors does not trim the internal capacitor values (UTEST Misc bit 5:1 = 00000b (XOSC LOAD CAP SEL)).

The internal load capacitor values stored in UTEST flash memory row have triple-voting flip-flop (TVF) implementation to prevent an incorrect value, and thus an undetectable error in the system.

The XOSC has the ability to be started with either 8 MHz to 20 MHz, or 40 MHz to 44 MHz crystal. The XOSC EN_40MHz bit in the UTEST Misc row selects which crystal source will be used. The default value of this field is for an 8 MHz to 20 MHz crystal (UTEST Misc bit 10 = 0).

See the flash memory chapter for more information on the UTEST row programming.

27.5.2.2 XOSC register write protection

The XOSC registers with write protection are defined in [Table 346](#). The protected registers have write protection with both soft and hard locking. A soft lock can be unlocked by software after being previously locked. A hard lock is only unlocked by a reset once locked. The REG_PROT module is used to implement the XOSC register write protection.

Table 346. XOSC register write protection⁽¹⁾

Offset	Register	Protections?
00h	XOSC_CTL—XOSC Control Register	Yes

1. See Register Protection Configuration chapter for bit field details

27.5.2.3 XOSC reset value

[Table 347](#) shows the default reset value for the XOSC registers.

Table 347. XOSC register reset values

Offset	Register	Reset value
00h	XOSC_CTL—XOSC Control Register	8030_8000h

27.5.3 16 MHz internal RC oscillator (IRCOSC)

The microcontroller has a 16 MHz internal RC oscillator that is always enabled and can be used as the clock source for the PLLs. The IRCOSC is the default clock after reset. The register interface is used for user trimming of the oscillator and dividing the output from 1 to 32.

27.5.3.1 IRCOSC register interface

This device has a dedicated digital interface for the IRCOSC. The digital interface provides a register for fine tuning the IRCOSC frequency by the user. Other registers allow reading of the settings of IRCOSC temperature sensor, voltage regulator and capacitor trimming values that are set by the factory. The IRCOSC is always enabled when the device is powered.

27.5.3.2 IRCOSC reset value

[Table 348](#) shows the default reset value for the IRCOSC registers. The values of the IRCOSC registers are loaded from TEST DCF flash memory (16 MHz RCOSC_1, 16 MHz RCOSC_2, 16 MHz RCOSC_3, 16 MHz RCOSC_4) (see Device Configuration Format Records chapter for details).

Table 348. IRCOSC register reset values

Offset	Register	Reset value
04h	NT—IRCOSC Native Trimming Register	0000_0000h
08h	TT—IRCOSC Temperature Trimming Register	0000_0000h

27.5.3.3 IRCOSC register write protection

The IRCOSC registers do not utilize write protection.

27.6 Peripheral clocks

[Figure 261](#) shows the clock distribution to the cores and peripheral modules.

27.6.1 HSM clock divider

The clock dividers for the HSM core and random number generator (TRNG) are located outside the HSM, but not in the MC_CGM. These dividers are controlled by the HSM.

The HSM contains an internal PIT/RTI block, which is clocked by the IRCOSC.

The HSM has an internal clock monitor on its input clock, so there is no need for a Clock Monitoring Unit (CMU) for the HSM_CLK in [Figure 258](#).

27.6.2 PSI5 clock dividers

The SPC57EM80xx microcontroller includes three PSI5 modules, all of which use the three common clocks shown in [Figure 258](#). The required frequencies for these clocks are generated from PLL0 as follows:

- $F_{\text{PSI5_f189_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC0}[\text{DIV}] + 1) / 10 \text{CGM_AC5_DC0}[\text{FMT}]) = 6.048 \text{ MHz} \pm 0.1\%$
- $F_{\text{PSI5_f125_CLK}} = F_{\text{PLL0_PHI}} / (\text{CGM_AC5_DC1}[\text{DIV}] + 1) = 4 \text{ MHz}$
- $F_{\text{PSI5_1us_CLK}} = F_{\text{PLL0_PHI}} / (\text{CGM_AC5_DC2}[\text{DIV}] + 1) = 1 \text{ MHz}$

The nine PSI5-S clocks are generated by dividing the PSI5_1μs_CLK further via a second and a third level of 'cascaded' clock dividers. The individual frequencies are, therefore, derived from the PLL0 PHI frequency as follows:

- $F_{\text{PSI5_S_TS_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC0}[\text{DIV}] + 1))$
- $F_{\text{PSI5_S_DDSR0_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC1}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC10}[\text{DIV}] + 1))$
- $F_{\text{PSI5_S_DDSR1_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC1}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC11}[\text{DIV}] + 1))$
- $F_{\text{PSI5_S_DDSR2_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC1}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC12}[\text{DIV}] + 1))$
- $F_{\text{PSI5_S_DDSR3_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC1}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC13}[\text{DIV}] + 1))$
- $F_{\text{PSI5_S_WD0_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC20}[\text{DIV}] + 1))$
- $F_{\text{PSI5_S_WD1_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC21}[\text{DIV}] + 1))$
- $F_{\text{PSI5_S_WD2_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC22}[\text{DIV}] + 1))$
- $F_{\text{PSI5_S_WD3_CLK}} = F_{\text{PLL0_PHI}} / ((\text{CGM_AC5_DC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC2}[\text{DIV}] + 1) * (\text{CGM_AC5_CDC23}[\text{DIV}] + 1))$

27.6.3 LFAST clocking

The SPC57EM80xx includes two LFAST modules, one to support interprocessor communications in conjunction with the SIPI module, and one to support high speed debug in conjunction with the JTAGM module. A single LFAST PLL that requires a 10–20 MHz reference supports high speed operation of both LFAST modules. For low speed LFAST operation, the reference clock is used directly by the LFAST modules.

The LFAST PLL reference clock source may be the PLL0:PHI output, external oscillator (XOSC) or input from the external LFAST device via the DRCLK_IO package pin. When this reference clock is generated internally, it can be output on the DRCLK_IO pin.

The debug LFAST module only supports the operation mode where it outputs the reference clock to a connected LFAST capable tool, as the DRCLK pin only supports output of the DRCLK reference.

On Emulation and Debug Devices (ED), the LFAST module on the Buddy Die (BD) is used and connected to the package debug pins instead of the Production Die (PD) debug LFAST module. This module has a separate dedicated LFAST PLL, although as with non EDs', the DRCLK pin only supports output of the reference clock to a connected LFAST capable tool.

[Figure 262](#) shows the clock routing for the various LFAST modules on the device, including connections to the DRCLK_IO and DRCLK pins.

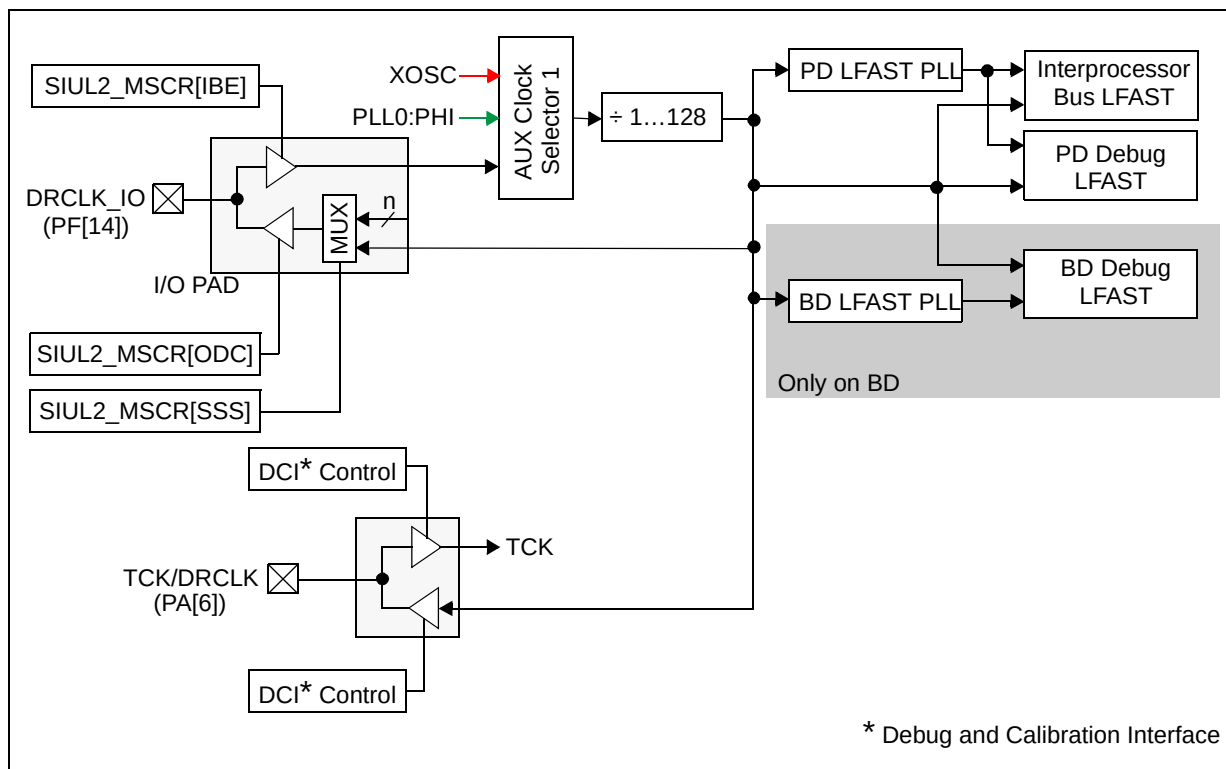


Figure 262. Device LFAST clocking

27.6.4 Ethernet clocking

Both internal and external clocks sources for both MII and RMII Ethernet interface modes are supported. The MII clocks (FEC_TXCLK/FEC_RXCLK) are 25 MHz, and the RMII clock (FEC_REF_CLK) is 50 MHz. [Figure 263](#) gives the connections for the Ethernet clocks.

The RXCLK pin is an input, used in MII mode only, and ignored by the RMII gasket in RMII mode.

The FEC_TXCLK pin is an input, used in MII mode only, but is shared with the FEC_REF_CLK input of the RMII gasket. Therefore, the Aux clock selector is required to select between FEC_TXCLK and FEC_REF_CLK for MII and RMII modes.

The FEC_REF_CLK can be generated internally or input to the device. For MII mode, the FEC_REF_CLK can be used for the 25 MHz reference for the external Ethernet PHY, which provides the TXCLK/RXCLK clocks back to the device. For RMII mode, FEC_REF_CLK can

be generated internally and fed to both the PHY device and to the RMI1 gasket by enabling the FEC_REF_CLK pad input and output buffers. FEC_REF_CLK can also be provided externally as an input to the device.

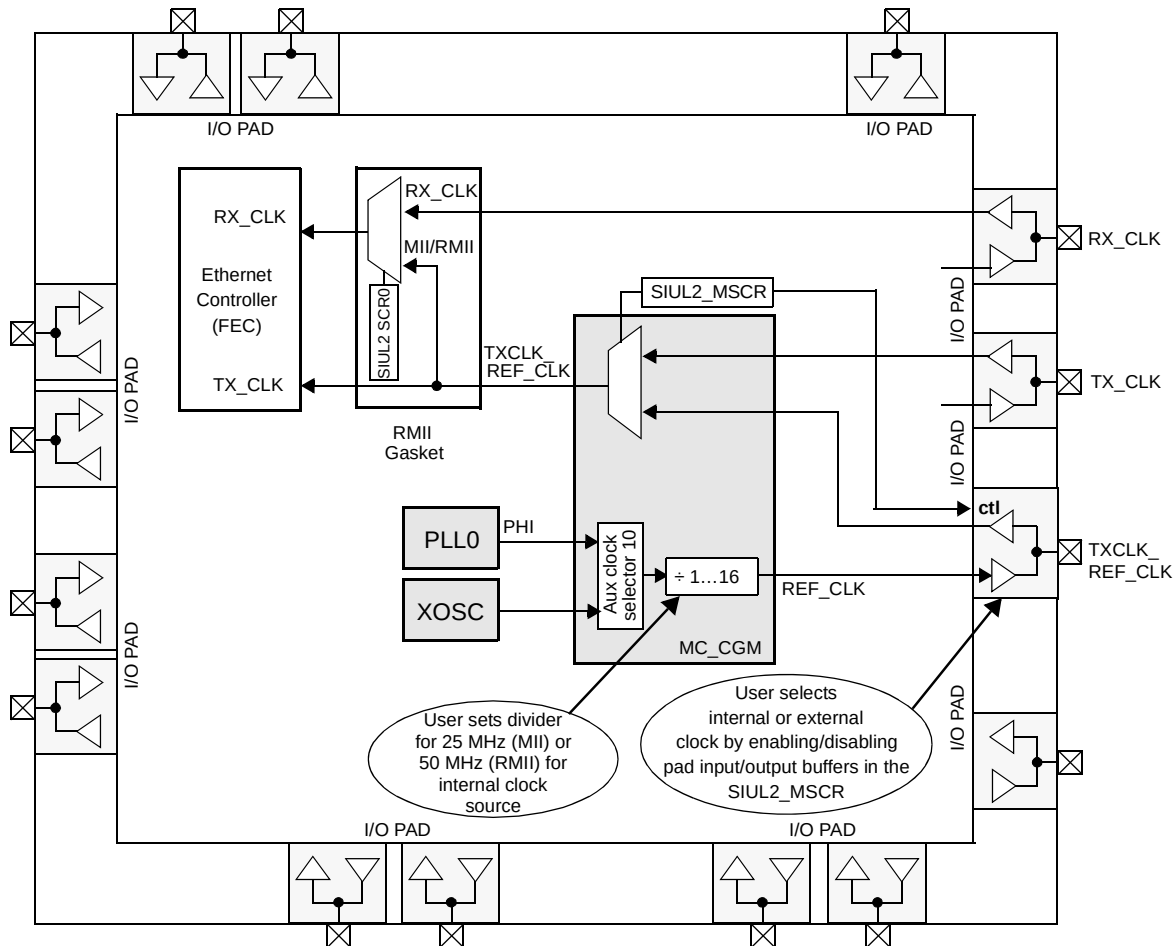


Figure 263. Device Ethernet clocking

27.6.5 FlexRay clocking

The FlexRay protocol clock is sourced from either PLL0:PHI clock or the external oscillator (XOSC), but its source selection is different from other modules. PLL0:PHI enters the MC_CGM where it is divided by a value that is configured in the CGM_AC2_DC0. The output of the divider is the FRAY_PLL_CLK, which then passes through the Pseudo Random Clock Divider (PRCD). This module is passive in normal operation and has no impact on the clock frequency, but when enabled it divides the FRAY_PLL_CLK by 16 and adds approximately 40% of random jitter to both edges of the clock. The PHI output of PLL0 is the source whenever the PRCD is enabled.

The FRAY_CLK is selected for the FlexRay block via an internal multiplexer. Either the XOSC or the output of the PRCD can be used as the FRAY_CLK. When using the XOSC, a 40 MHz crystal is required for proper operation. When a 40MHz oscillator is used as the input clock to the FlexRay controller, both edges of the clock are used by the module to

achieve the required 80 MHz frequency. Selection of PLL or external oscillator as the FlexRay clock is configured in the MC_CGM (see [Figure 350 \(MC_CGM auxiliary clock 2 generation overview\)](#) in [Chapter 30: Clock Generation Module \(MC_CGM\)](#)).

27.6.6 M_TTCAN/M_CAN clocking

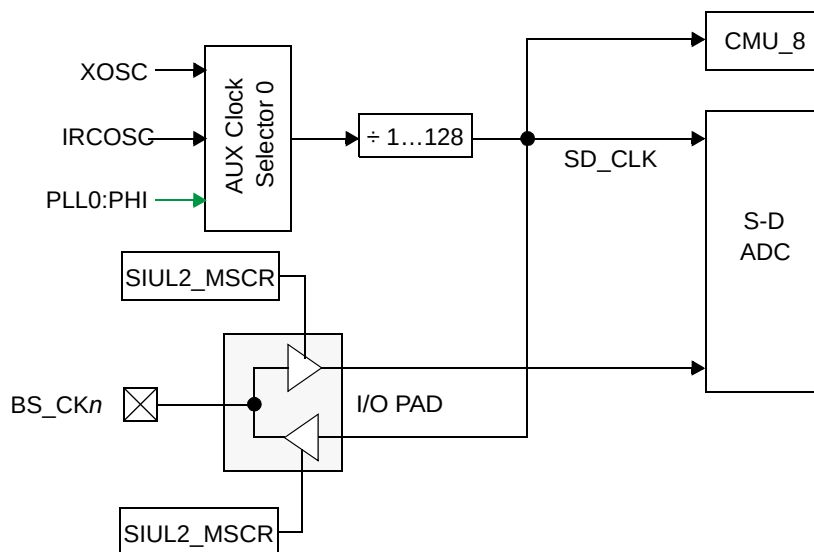
The CAN_CLK is generated from either the XOSC or PLL0:PHI clocks, depending on the configuration of CGM_AC8_SC in the MC_CGM. The CAN_CLK can also be reduced by configuring the dividing factor in the CGM_AC8_DC0 (divide by 1 to 64). The CAN_CLK then passes through the PRCD. This module is passive in normal operation and has no impact on the clock frequency, but when enabled it divides the CAN_CLK by 16 and adds approximately 40% of random jitter to both edges of the clock. The PHI output of PLL0 is the source whenever the PRCD is enabled.

27.6.7 Sigma-Delta ADC clocking

The analog portion of the Sigma-Delta ADC (SDADC) works with either an internal or external modulator. The internal clock (SD_CLK, see [Figure 261](#)) is always used when the internal modulator in the SDADC is enabled. This device supports either an external input clock (BS_CK), or internally generated output clock (SD_CLK) when using an external modulator. The output clock is generated internally and connected to the external modulator clock package pin (BS_CK). There is a dedicated external modulator clock pin for each SDADC.

The SDADC contains an internal mux for selection between the internal and external clocks. The internal clock is selected when the internal modulator is enabled, and the external clock is selected when it is disabled. In order to provide an internal clock option when the internal modulator is enabled, the SD_CLK output from the MC_CGM must be connected to the BS_CK pin as shown in [Figure 264](#). Control of the input/output direction of the BS_CK pin is handled by writing the MSCR in the SIUL2. Each SDADC is connected as shown in [Figure 264](#), with the SD_CLK being common to all modules, but each SDADC having a dedicated BS_CK pin.

Figure 264. Sigma-Delta ADC clocking



27.6.7.1 System Clock/GTM output clock/Ethernet clock pin muxing

There are two system clock output pins on the device, SYSCLK[1:0]. The output clock from either of the PLLs, the internal oscillator (IRCOSC), or external oscillator (XOSC) can be individually selected for either SYSCLK[1:0]. See [Figure 258](#).

There are three GTM output clocks available on package pins, EXTCLK[2:0]. EXTCLK[1:0] are muxed on the same pins as SYSCLK[1:0]. EXTCLK[2] is muxed on the same pin as the Ethernet RMII clock (FEC_REF_CLK). The selection of EXTCLK[n], SYSCLK[n] or FEC_REF_CLK is done in the SIUL2_MSCR registers and shown in [Figure 265](#).

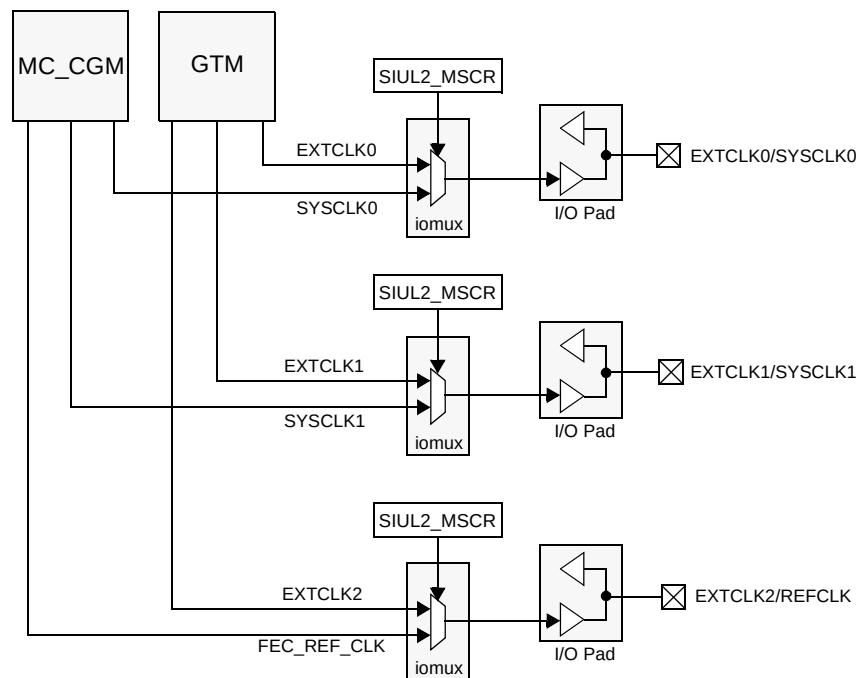


Figure 265. SYSCLK/EXTCLK/REF_CLK pin muxing

27.7 Clock monitoring

For all safety relevant clocks the microcontroller uses clock monitoring units (CMUs) to detect a missing clock or incorrect frequency. Each CMU is programmed independently (see [Figure 258](#) for CMU distribution). The IRCOSC and XOSC are used as the clock monitor references. Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

27.7.1 CMU configuration

This section explains the CMU configuration.

[Figure 266](#) shows the block diagram for CMU0 on the SPC57EM80xx.

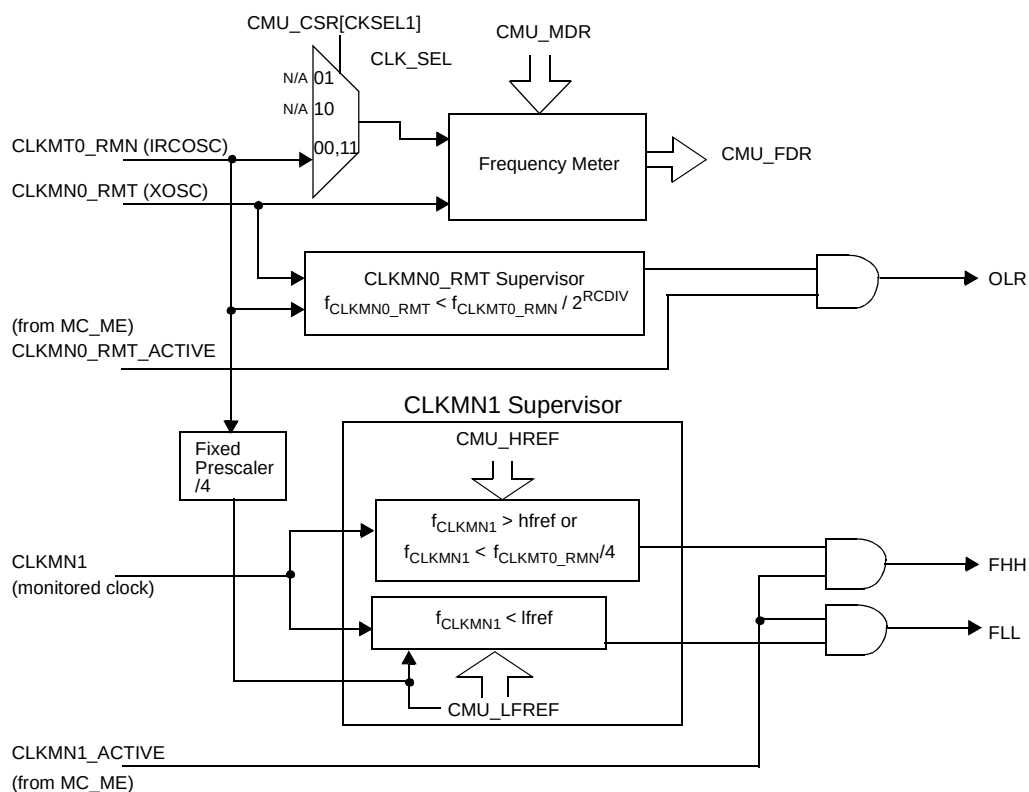


Figure 266. CMU0 block diagram

Figure 267 shows the block diagram for CMU[1:10] on the SPC57EM80xx.

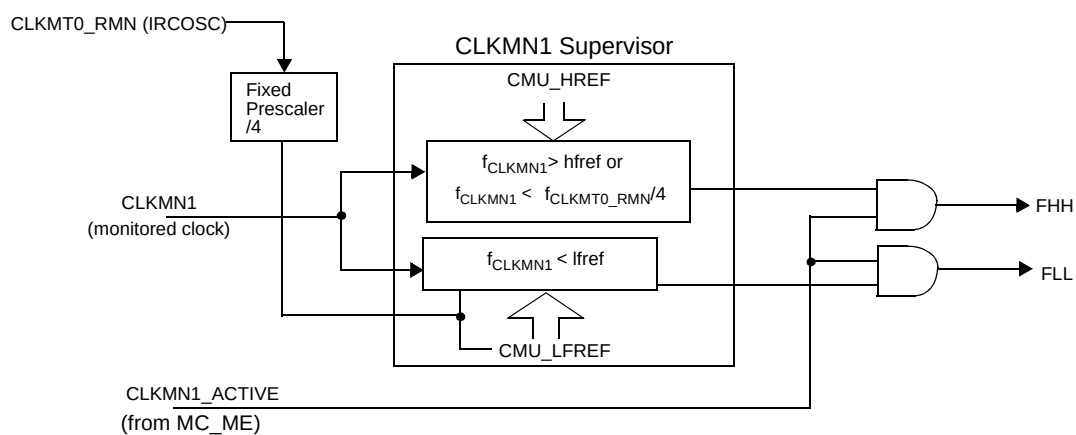


Figure 267. CMU[1:10] block diagram

27.7.1.1 Clock input sources

[Table 349](#) shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRCOSC is the reference clock for all clock monitors. Only CMU0 implements the XOSC monitor. CMU0 uses the IRCOSC clock to measure if the XOSC is too low. CMU0 can also be used to calibrate the IRCOSC frequency using the XOSC. All other CMUs are configured identically.

Table 349. Clock input sources

Clock module	Monitored clock
CMU0	PLL0:PHI, XOSC, IRCOSC
CMU1	Peripheral Core_2, Main Core_1, Safety Core (Main Core_0, Checker Core_0s), FXBAR_CLK
CMU2	SXBAR_CLK
CMU3	PBRIDGE_CLK
CMU4	PER_CLK
CMU5	SD_CLK
CMU6	SAR_CLK
CMU7	SENT_CLK
CMU8	PS15_f189_CLK
CMU9	PS15_f125_CLK
CMU10	PS15_1μs_CLK

27.7.1.2 CMU registers and field availability

[Table 350](#) specifies which registers and fields are available on a given CMU for this device.

Table 350. CMU registers availability

Address offset	Register	CMU	Note
0000h	CMU_CSR	All	CMU_CSR[SFM] and CMU_CSR[CLSEL1] are in CMU0 only.
0004h	CMU_FDR	CMU0	—
0008h	CMU_HFREFR	All	—
000Ch	CMU_LFREFR	All	—
0010h	CMU_ISR	All	CMU_MDR[OLRI] are in CMU0 only.
0014h	Reserved	—	—
0018h	CMU_MDR	CMU0	—

27.7.1.3 CMU register write protection

The CMU registers have write protection as shown in [Table 346](#). A soft lock can be unlocked by software after being previously locked. A hard lock is only unlocked by a reset once locked. The REG_PROT module is used to implement the CMU register write protection.

Table 351. CMU register write protection

Offset	Register ⁽¹⁾	Protections?
0000h	CMU Control Status Register (CMU_CSR)	Yes
0004h	CMU Frequency Display Register (CMU_FDR)	No
0008h	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	No
000Ch	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	No
0010h	CMU Interrupt Status Register (CMU_ISR)	No
0014h	Reserved	No
0018h	CMU Measurement Duration Register (CMU_MDR)	No

1. See Register Protection chapter for bit field details.

27.7.2 PLL0 monitor

Software can program an upper and lower limit on the expected PLL0 PHI output clock frequency. If the monitor is enabled and the measured frequency is above or below the limits, a flag bit is set and an interrupt, if enabled, is generated. The default condition of the clock monitor is disabled.

27.7.3 External oscillator (XOSC) monitor

The XOSC frequency is compared to a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set and an interrupt, if enabled, is generated.

27.7.4 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

27.7.5 System clock monitors

A CMU is assigned to monitor the frequency of the computational core, IOP core, fast and slow XBAR, both peripheral bridges, peripheral, PSI5, SENT and ADC clocks (refer to [Figure 258](#)). The HSM does not have an assigned CMU since it has its own internal clock monitor.

27.8 Loss of system clock behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and provides several options for reaction to a loss of clock in the application. [Figure 268](#) gives a high-level view of the loss of clock logic.

27.8.1 Loss of PLL/XOSC clock

As shown in [Figure 268](#), each loss of lock signal from each PLL and the XOSC failure signals are monitored by the FCCU. The FCCU can be programmed to generate a short or long reset sequence when a clock failure occurs, or simply to generate an interrupt. In the case of the interrupt, only the backup clock in the PLL can provide clocks to the system in

order to perform a system clock switch. There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).

If a long reset is selected the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states, and the system clock is switched to the IRCOSC. If a short reset is selected, the clock configuration is maintained, but again, the PLL backup clock is the only available clock to clock the system through an MC_ME mode switch.

27.8.2 Loss of IRCOSC clock

The frequency of the IRCOSC clock is monitored by the frequency meter in CMU_0. There is no automated trigger of an FCCU error condition if the IRCOSC fails. Since the IRCOSC is the boot and backup clock, a failure is a catastrophic failure.

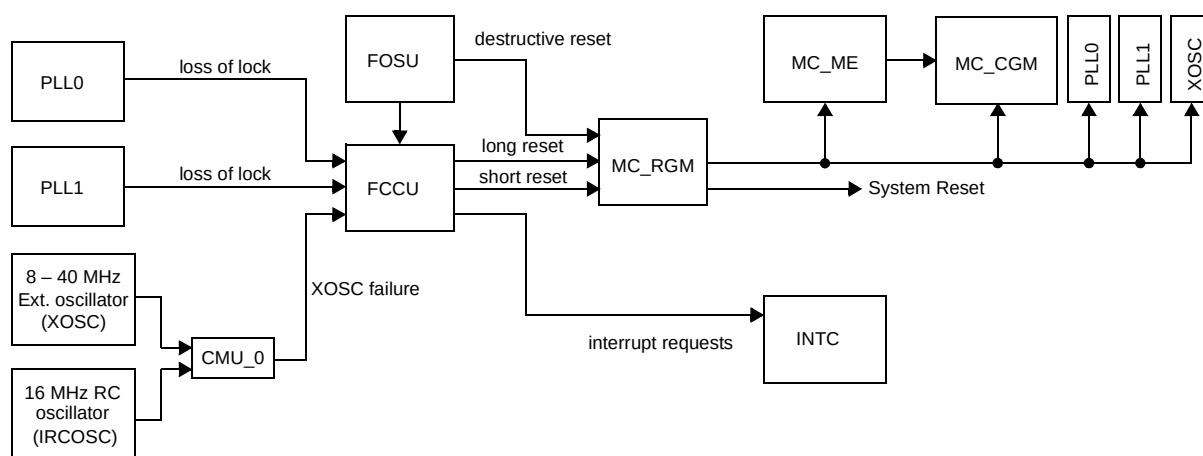


Figure 268. Loss of clock logic

28 Dual PLL Digital Interface (PLLDIG)

28.1 Introduction

The MCU provides a user interface and control over the dual PLL system composed of PLL0 and PLL1 blocks (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1 as shown in the “Clocking” chapter. The key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by the PLL1 PHI frequency modulated output.

28.2 Block Diagram

The block diagram of the PLLDIG block is found in the “Clocking chapter” of the *SPC57EM80xx Reference Manual*.

28.3 Features

The dual PLL has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for reducing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Input clock frequency range from 8 MHz to 56 MHz before the prescaler, and output clock from 4.762 MHz to 625 MHz for PLL0.
- Input clock frequency range from 37.5 MHz to 78.125 MHz, and output clock from 4.762 MHz to 625 MHz for PLL1.
- Programmable frequency modulation on PLL1.
- Lock detect circuitry reports when PLL0 and PLL1 achieves frequency lock, and continuously monitors lock status to report loss of lock conditions.
 - User-selectable ability to generate an interrupt request upon loss of lock.
 - User-selectable ability to generate a system reset upon loss of lock.
 - [BLG_PLLDIG_003][Covers: ADD12.023][Covers: Saf3005]The loss of lock indication is sent to the FCCU via the system glue logic. [end]

28.4 Modes of operation

The operating modes of the PLLs is determined by the value of PLLnCR[CLKCFG]. The various modes of operation for the PLL are:

- Normal mode with reference, and either PLL0 or both PLLs enabled

Note: .Mode changes are controlled by configuring the MC_ME Mode Configuration registers (MC_ME_<mode>_MC).

Normal mode is defined as the mode where the clocks are driven by the PLL. When using a crystal for the clock reference, reset should remain asserted until the oscillator has stabilized.

28.4.1 Normal mode with reference, PLL0 or both PLLs enabled

In the normal mode, PLL0 receives an input clock from the reference and the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0_PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, output divider (Reduced Frequency Divider) and whether the PLL1 is modulating or not can be programmed by writing to the PLL1FM register, [Section 28.5.2.7, PLL1 Frequency Modulation Register \(PLL1FM\)](#).

Note: While powering down the PLLs, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Only when the PLL0 achieves lock should PLL1 be configured to work in either mode by programming the appropriate MC_ME_<mode>_MC register.

28.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs.

28.5.1 Memory map

[Table 352](#) shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

Table 352. Dual PLL Digital Interface memory map

Offset (hex)	Register	Access	Reset Value (hex)	Section/Page
0000	PLL0CR — PLL0 Control Register	R/W	0000 0000h	on page 899
0004	PLL0SR — PLL0 Status Register	R/W	0000 0000h	on page 900
0008	PLL0DV — PLL0 Divider Register	R/W	zzzz z0zzh ⁽⁷⁾	on page 901
000C–001C	Reserved	—	—	—
0020	PLL1CR — PLL1 Control Register	R/W	0000 0000h	on page 903
0024	PLL1SR — PLL1 Status Register	R/W	0000 0000h	on page 904
0028	PLL1DV — PLL1 Divider Register	R/W	zzzz z0zzh ⁽⁷⁾	on page 905
002C	PLL1FM — PLL1 Frequency Modulation Register	R/W	0000 0000h	on page 906
0030	PLL1FD — PLL1 Fractional Divide Register	R/W	0000 0000h	on page 908
0034–0038	Reserved	—	—	—

7. See the “Clocking” chapter for default reset values.

28.5.2 Register descriptions

Some of the register reset values are specifically configured for each unique device by external configuration signals or parameters.

28.5.2.1 PLL0 Control Register (PLL0CR)

Figure 269. PLL0 Control Register (PLL0CR)

Offset: 0000h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CLKCFG ⁽¹⁾		EXPDIE	0	0	0	LOLIE	LOLRE	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_<mode>_MC register.

Table 353. PLL0CR field descriptions

Field	Description
0:21	Reserved
22:23 CLKCFG	Clock Configuration. This field indicates the operational state of PLL. 00PLL off. 01Reserved 10Reserved. 11Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b. <i>Note: In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.</i>
24 EXPDIE	External Power Down Cycle Complete indication interrupt enable. This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
25	Reserved
26	Reserved
27	Reserved

Table 353. PLL0CR field descriptions (continued)

Field	Description
28 LOLIE	Loss of lock interrupt enable. The LOLIE bit enables a loss of lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0Ignore loss of lock. Interrupt not requested. 1Enable interrupt request upon loss of lock.
29 LOLRE	Loss of lock reset enable. The PLL0CR[LOLRE] bit determines whether system reset is asserted or not upon a loss of lock indication from PLL. If both PLL0CR[LOLRE] = 1 and PLL0CR[LOLIE] = 1, PLL0CR[LOLRE] has precedence and the device asserts reset with the interrupt not being taken. 0Ignore loss of lock. Reset not asserted. 1Assert reset on loss of lock when operating in normal mode.
30	Reserved
31	Reserved

28.5.2.2 PLL0 Status Register (PLL0SR)

Figure 270. PLL0 Status Register (PLL0SR)

Offset: 0004h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Table 354. PLL0SR field descriptions

Field	Description
0:23	Reserved
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag. This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
25	Reserved

Table 354. PLL0SR field descriptions (continued)

Field	Description
26	Reserved
27	Reserved
28 LOLF	Loss of lock flag. This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0No loss of lock detected. Interrupt service not requested. 1Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. 0PLL is unlocked. 1PLL is locked.
30	Reserved
31	Reserved

28.5.2.3 PLL0 Divider Register (PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider. The values of PREDIV and MFD should not be changed when the PLL is on and locked (working in Normal mode). If these fields are changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled. The reduced frequency divider fields (RFDPHI, RFDPHI1) can be modified at anytime, but the changes only become effective after PLL0 is disabled, then reenabled.

Figure 271. PLL0 Divider Register (PLL0DV)

Offset: 0008h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RFDPHI1				0	0	0	0	0	RFDPHI					
W																
Reset	0	_(1)	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PREDIV				0	0	0	0	0	MFD					
W																
Reset	0	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. See the Clocking chapter for reset value information.

Table 355. PLL0DV field descriptions

Field	Description
0	Reserved
1:4 RFDPHI1	PHI1 reduced frequency divider. This 4-bit field determines the VCO/2 clock post divider for driving the PHI1 output clock. 00xx Reserved 0100 Divide by 4 0101 Divide by 5 1110 Divide by 14 1111 Divide by 15
5:9	Reserved
10:15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 00 Reserved 01 Divide by 2 02 Divide by 4 03 Divide by 6 3E Divide by 124 3F Divide by 126
16	Reserved
17:19 PREDIV	Input clock pre-divider. This 3-bit field controls the value of the divider on the input clock. The output of the pre-divider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 110 Divide by 6 111 Divide by 7
20:24	Reserved
25:31 MFD	Loop multiplication factor divider. This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 00 Reserved 07 Reserved 08 Divide by 8 7E Divide by 126 7F Divide by 127

28.5.2.4 PLL1 Control Register (PLL1CR)

Figure 272. PLL1 Control Register (PLL1CR)

Offset: 0020h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CLKCFG ⁽¹⁾		EXPDIE	0	0	0	LOLIE	LOLRE	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_<mode>_MC register.

Table 356. PLL1CR field descriptions

Field	Description
0:21	Reserved
22:23 CLKCFG	Clock Configuration. This field indicates the operational state of PLL. 00PLL off. 01Reserved 10Reserved. 11Normal mode with PLL running. When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b. <i>Note: In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.</i>
24 EXPDIE	External Power Down Cycle Complete indication interrupt enable. This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
25	Reserved
26	Reserved
27	Reserved
28 LOLIE	Loss of lock interrupt enable. The LOLIE bit enables a loss of lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0Ignore loss of lock. Interrupt not requested. 1Enable interrupt request upon loss of lock.

Table 356. PLL1CR field descriptions (continued)

Field	Description
29 LOLRE	Loss of lock reset enable. The PLL1CR[LOLRE] bit determines whether system reset is asserted or not upon a loss of lock indication from PLL. If both PLL1CR[LOLRE] = 1 and PLL1CR[LOLIE] = 1, PLL1CR[LOLRE] has precedence and the device asserts reset with the interrupt not being taken. 0Ignore loss of lock. Reset not asserted. 1Assert reset on loss of lock when operating in normal mode.
30	Reserved
31	Reserved

28.5.2.5 PLL1 Status Register (PLL1SR)

Figure 273. PLL1 Status Register (PLL1SR)

Offset: 0024h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Table 357. PLL1SR field descriptions

Field	Description
0:23	Reserved
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag. This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
25	Reserved
26	Reserved
27	Reserved

Table 357. PLL1SR field descriptions (continued)

Field	Description
28 LOLF	Loss of lock flag. This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0No loss of lock detected. Interrupt service not requested. 1Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. 0PLL is unlocked. 1PLL is locked.
30	Reserved
31	Reserved

28.5.2.6 PLL1 Divider Register (PLL1DV)

The PLL1DV register provides the output clock reduced frequency dividers and loop divider values. The value of MFD should not be changed when the PLL is on and locked (working in Normal mode). If the field is changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled. The reduced frequency divider field (RFDPHI) can be modified at anytime, but the changes only become effective after PLL1 is disabled, then reenabled .

Figure 274. PLL1 Divider Register (PLL1DV)

Offset: 0028h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	RFDPHI ⁽¹⁾					
W																
Reset	0	0	0	0	0	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	MFD ⁽¹⁾						
W																
Reset	0	0	0	0	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. See the Clocking chapter for reset value information.

Table 358. PLL1DV field descriptions

Field	Description
0	Reserved
10:15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 00 Reserved 01 Divide by 2 02 Divide by 4 3E Divide by 124 3F Divide by 126
16	Reserved
25:31 MFD	Loop multiplication factor divider. This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 10 Divide by 16 11 Divide by 17 21 Divide by 33 22 Divide by 34 All other settings are Reserved

28.5.2.7 PLL1 Frequency Modulation Register (PLL1FM)

The PLL1FM register enables frequency modulation on PLL1 and provides controls for the FM spread spectrum, modulation depth and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of PLL1PLL2FM[MODEN] and PLL1PLL2FM[MODSEL] fields once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (if PLL1CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality. The modulation period (PLL1FM[MODPRD]) and increment step fields (PLL1FM[INCSTP]) can be changed without losing lock, however, the output clock from PLL may have an incorrect frequency for a few cycles after either of these updates (as defined in the datasheet).

Figure 275. PLL1 Frequency Modulation Register (PLL1FM)

Offset: 002Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MODEN	MODSEL		MODPRD											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	INCSTP														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 359. PLL1FM field descriptions

Field	Description
0	Reserved
1 MODEN	Modulation enable. This bit enables the frequency modulation. 0Frequency modulation disabled. 1Frequency modulation enabled.
2 MODSEL	Modulation selection. This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0Modulation centered around nominal frequency. 1Modulation spread below nominal frequency.
3:15 MODPRD	Modulation period. This 13-bit field is the binary equivalent of the modulation period variable derived from the formula: $\text{MODPRD} = \text{round}\left(\frac{f_{\text{ref}}}{4 \times f_{\text{mod}}}\right)$ where f_{ref} represents the frequency of the feedback divider and f_{mod} represents the modulation frequency.
16	Reserved
17:31 INCSTP	Increment step. This 15-bit field is the binary equivalent of the INCSTP variable derived from the formula: $\text{INCSTEP} = \text{round}\left(\frac{(2^{15} - 1) \times md \times \text{PLL1CR}[\text{MFD}]}{100 \times 5 \times \text{MODPRD}}\right)$ where md represents the peak modulation depth in percentage (+/- md for centered modulation, - $2 \times md$ for modulation below nominal frequency), and MFD represents the nominal value of the feedback loop divider.

28.5.2.8 PLL1 Fractional Divider Register (PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (if PLL1CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality. The dither disable and fractional divider fields can be changed without losing lock, however, the output clock from PLL can have an incorrect frequency for a few cycles after either of these updates.

Figure 276. PLL1 Fractional Divider Register (PLL1FD)

Offset: 0030h										Access: User read/write							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0	FDEN	0	0	0	0	0	0	0	0	0	0	0	0	DTHDIS		
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	FRCDIV												
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 360. PLL1FD field descriptions

Field	Description
0	Reserved
1 FDEN	Fractional Divide Enable. This bit enables the fractional divider in the loop divider for PLL1. 0Fractional divide disabled. 1Fractional divide enabled.
2:13	Reserved
14:15 DTHDIS	Dither Disable. See chip specific chapter for more details on dither.
16:19	Reserved
20:31 FRCDIV	Fractional divide input. When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the VCO clock frequency is given by: $\text{VCO Frequency} = \text{InputClock} \times (\text{PLL1DV}[\text{MFD}] + (\text{FRCDIV} \div 2^{12}))$ where FRCDIV is the decimal equivalent of the FRCDIV bits.

28.5.3 Register classification for safety requirements

This module is classified as a safety Vital Module (ViMo). Hence, the registers of this module have been classified as shown in the SPC57EM80xx Reference Manual “Functional Safety” chapter.

28.6 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

28.6.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for each PLL are discussed in detail in the *SPC57EM80xx Data Sheet*.

28.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV and PLL1DV registers, and calculated according to the following equations:

Equation 4

$$f_{\text{pll0_phi}} = f_{\text{pll0_ref}} \times \frac{\text{PLL0DV}[\text{MFD}]}{\text{PLL0DV}[\text{PREDIV}] \times \text{PLL0DV}[\text{RFDPHI}]}$$

Equation 5

$$f_{\text{pll0_phi1}} = f_{\text{pll0_ref}} \times \frac{\text{PLL0DV}[\text{MFD}]}{\text{PLL0DV}[\text{PREDIV}] \times \text{PLL0DV}[\text{RFDPHI1}] \times 2}$$

Equation 6

$$f_{\text{pll1_phi}} = f_{\text{pll1_ref}} \times \frac{\text{PLL1DV}[\text{MFD}]}{\text{PLL1DV}[\text{RFDPHI}]}$$

When PLL0s PHI1 is used as the reference for PLL1:

Equation 7

$$f_{\text{pll1_phi}} = \frac{f_{\text{pll0_ref}} \times \text{PLL0DV}[\text{MFD}] \times \text{PLL1DV}[\text{MFD}]}{\text{PLL0DV}[\text{PREDIV}] \times \text{PLL0DV}[\text{RFDPHI1}] \times \text{PLL1DV}[\text{RFDPHI}] \times 2}$$

The relationship between the VCO frequency f_{VCO} and the output frequency of the PLLs is determined by the value of PLL1DV[RFDPHI], PLL0DV[RFDPHI] and PLL0DV[RFDPHI1], according to the following equations:

Equation 8

$$f_{\text{pll0_VCO}} = f_{\text{pll0_phi}} \times \text{PLL0DV}[\text{RFDPHI}]$$

Equation 9

$$f_{pll1_vco} = f_{pll1_phi} \times PLL1DV[RFDPHI]$$

Equation 10

$$f_{pll0_vco} = f_{pll0_phi1} \times 2 \times PLL0DV[RFDPHI1]$$

Note: f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification for PLL0 or PLL1. Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLLnSR[LOCK] flag are immediately negated if the fields of PLLnDV are changed without powering down the analog PLLs.

Upon any of these events an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLLnSR[LOCK] flag is held negated. After the timer expires, the PLLnSR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLLnCR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

Note: The PLL must be powered down and powered up by configuring MC_ME_<mode>_MC before PLL0CR[PREDIV], PLL0CR[MFD], PLL1CR[MFD] or the input reference clock values are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in [Section 28.7, Initialization information](#).

28.6.3 Loss of lock

The PLL digital interface registers provide the flexibility to select whether to generate an interrupt assert system reset or do nothing in the event that the PLL loses lock according to the available PLLnCR options.

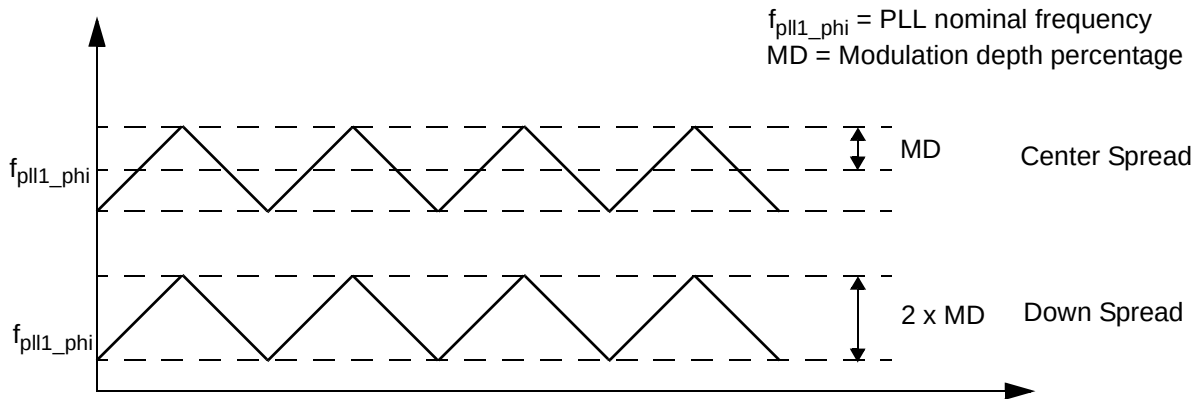
Loss of lock reset and interrupt are only generated when the PLL is operating in normal mode. The PLLDIG_PLLnSR[LOLF] bits are not asserted by a loss of lock condition detected during Bypass mode, although going to Bypass mode from normal mode does not automatically clear the flag if it was previously asserted while the PLL was in normal mode.

The lock indication from the analog PLL is synchronized and stored in the status register. When the analog PLL loses lock or regains lock, it will be immediately indicated in the status register (after synchronization time).

A pair of counters monitor the reference and feedback clocks to determine when the system has acquired frequency lock. Once the PLL has locked, the counters continue to monitor the reference and feedback clocks and report whether the PLL has lost lock. The lock status is indicated in the status register.

28.6.4 Frequency modulation

Frequency modulation uses a triangular profile as shown in [Figure 277](#). The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].



Note: The device maximum operating frequency includes the frequency modulation. If center modulation is selected, the fsys must be below fmax by MD percentage such that

Figure 277. Triangular frequency modulation

The following equations define how to calculate PLL1FM[MODPRD] ([Equation 11](#)) and PLL1FM[INCSTP] ([Equation 12](#)) based on the frequency of the feedback divider (f_{ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD):

Equation 11

$$PLL1FM[MODPRD] = \text{round}\left(\frac{f_{ref}}{4 \times f_{mod}}\right)$$

Equation 12

$$PLL1FM[INCSTP] = \text{round}\left(\frac{(2^{15} - 1) \times MD \times PLL1DV[MFD]}{100 \times 5 \times PLL1FM[MODPRD]}\right)$$

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

Equation 13

$$(PLL1FM[MODPRD] \times PLL1FM[INCSTP]) < 2^{15}$$

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is given by the following formula:

Equation 14

$$\text{ModulationDepth} = \text{round}\left(\frac{\text{PLL1FM}[\text{MODPRD}] \times \text{PLL1FM}[\text{INCSTP}] \times 100 \times 5}{(2^{15} - 1) \times \text{MFD}}\right)$$

As an example, suppose the following configuration:

- Input frequency: 4 MHz
- Loop divider (PLL n CR[MFD]): 64
- Input divider: 1
- VCO frequency: 4 MHz $\times$ 64 = 256 MHz
- PLL output frequency: 256 MHz / RFD
- Center spread (PLL1FM[MODSEL] = 0)
- Modulation frequency: 24 kHz
- Modulation depth: +/- 2.0% (4% peak-to-peak)
- PLL1FM[MODPRD] = Round[(4 $\times$ 10⁶)/(4 $\times$ 24 $\times$ 10³)] = Round[41.66] = 42
- PLL1FM[INCSTP] = Round[((2¹⁵ - 1) $\times$ 2 $\times$ 64) / (100 $\times$ 5 $\times$ 42)] = Round[199.722] = 200
- PLL1FM[MODPRD] $\times$ PLL1FM[INCSTP] = 42 $\times$ 200 = 8400 (which is less than 2¹⁵)
- MD (quantized) = ((42 $\times$ 200 $\times$ 100 $\times$ 5) / ((2¹⁵ - 1) $\times$ 64) = 2.00278%

In this example, the modulation error is 0.00278%.

The FM parameters can only be changed, and FM can only be enabled, when PLL0 is locked. Writing to the PLL1FM while the PLL0 is unlocked has no effect. Furthermore, when PLL0 loses lock, the FM modulation parameters are reset and modulation is disabled until PLL0 relocks and PLL1FM is reprogrammed. The sequence for reprogramming the FM is:

1. Power down PLL1^(f).
2. Program the PLL1FM while PLL1 is powered down.
3. Power up the PLL1^(g).

28.7 Initialization information

Coming out of reset PLL0 and PLL1 are enabled. The recommended procedure to program the PLLs and enter normal mode is:

1. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].
2. Provide a power down pulse (controlled by configuring the appropriate MC_ME_<mode>_MC register).
3. Poll PLL0SR[LOCK] until it asserts.
4. Program PLL1DV[MFD] and PLL1DV[RFDPHI].

f. By writing MC_ME_<mode>_MC[PLL1ON] = 0b.

g. By writing MC_ME_<mode>_MC[PLL1ON] = 1b.

5. If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.
6. Provide a power down pulse (controlled by configuring the appropriate MC_ME_<mode>_MC register).
7. Poll PLL1SR[LOCK] until it asserts.

Engage normal mode.

At this point, the PLL is configured and is generating a output clock; however, the PLL is not the system clock until it is configured in the Mode Entry Module (MC_ME) and a mode change has taken place. See the “Mode Entry Module (MC_ME)” chapter for more details.

29 Clock Monitor Unit (CMU)

29.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock sources CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and also detects if the monitored clock frequency leaves an upper or lower frequency boundary

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which in turn can then take whatever corrective action is required as configured in the FCCU.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the division factor given by the RCDIV field of CMU Control Status Register and reports this event. The CMU can also monitor CLKMN1 and generate events when the CLKMN1 frequency is less than the CLKMT0_RMN frequency divided by four or if CLKMN1 leaves an upper or lower frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

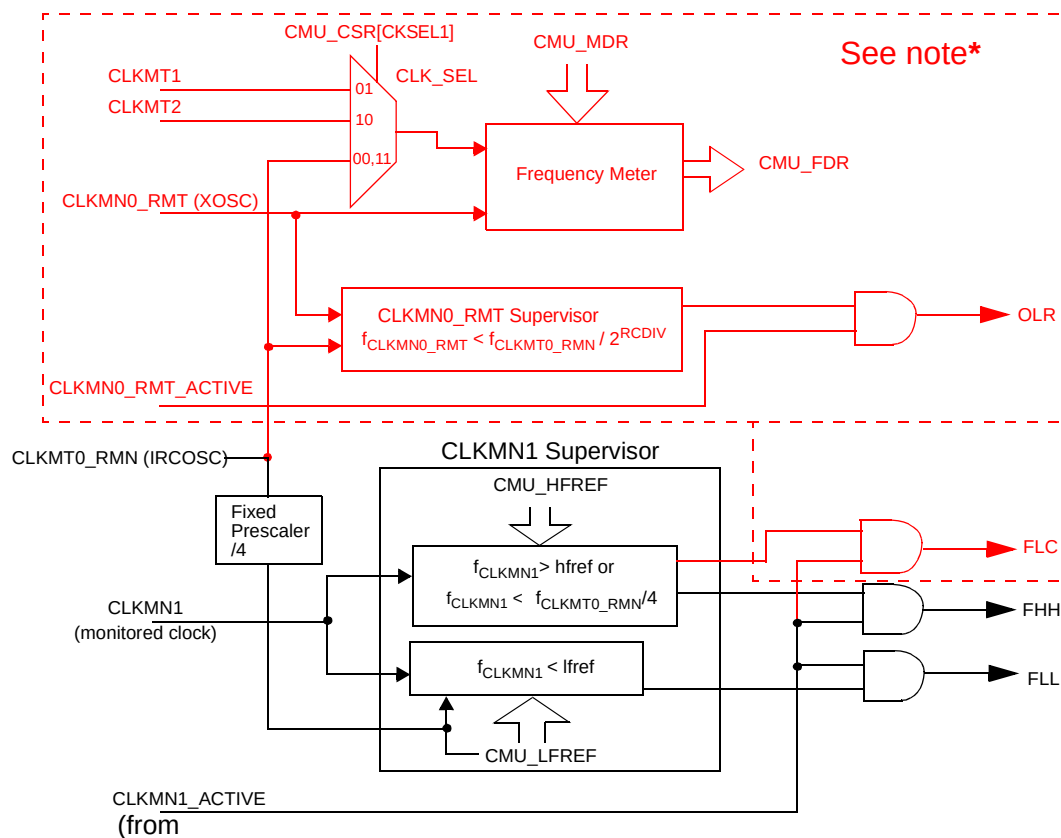
The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of one clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

Note: See the “Clocking” chapter for the number of instances of the CMU in this device.

29.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as the reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN/n clock.
- CLKMN1 frequency monitoring with respect to CLKMT0_RMN / 4 and upper or lower frequency boundary.
- Event generation for various failures detected inside monitoring unit.

29.2 Block diagram



*Not used in all CMU block configurations. See the “Clocking” chapter for specific CMU implementations.

Figure 278. Clock Monitor Unit diagram

29.3 Signals

Table 361 describes the signals on the boundary of the CMU (in alphabetical order).

Table 361. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference — Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.

Table 361. Signal description (continued)

Signal	I/O	Description
CLKMN1	I	Monitored Clock Signal 1 — Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference — Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

Note: See the “Clocking” chapter for device specific clock sources of the CMU.

29.4 Register description and memory map

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in [Table 362](#).

Table 362. CMU register summary

Offset (hex)	Register	Location
0000	CMU Control Status Register (CMU_CSR)	on page 917
0004	CMU Frequency Display Register (CMU_FDR)	on page 918
0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	on page 918
000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	on page 919
0010	CMU Interrupt Status Register (CMU_ISR)	on page 920
0014	Reserved	—
0018	CMU Measurement Duration Register (CMU_MDR)	on page 921

Note: See the “Clocking” chapter for register and field availability details.

29.4.1 Register descriptions

29.4.1.1 CMU Control Status Register (CMU_CSR)

Figure 279. CMU Control Status Register (CMU_CSR)

Address: Base + 0000h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SFM	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CKSEL1		0	0	0	0	0	RCDIV ⁽¹⁾		CME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details.

Table 363. CMU_CSR field descriptions

Field	Description
0:7	Reserved
8 SFM	Start frequency measure. The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR register. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9:21	Reserved
22:23 CKSEL1	Frequency measure clock selection bit. CKSEL1 selects the clock to be measured by the frequency meter. This only effects CMU instances that utilizes clock metering. 00 CLKMT0_RMN is selected 01 Reserved 10 Reserved 11 CLKMT0_RMN is selected
24:28	Reserved

Table 363. CMU_CSR field descriptions (continued)

Field	Description
29:30 RCDIV	CLKMTO_RMN division factor. These bits specify the CLKMTO_RMN division factor. The output clock frequency is f_{CLKMTO_RMN} divided by the factor 2^{RCDIV} . This output clock is used as reference clock to compare with CLKMNO_RMT for crystal clock monitor feature. The clock division coding is as follows: 00 CLKMTO_RMN divided by 1 (No division) 01 CLKMTO_RMN divided by 2 10 CLKMTO_RMN divided by 4 11 CLKMTO_RMN divided by 8
31 CME	CLKMN1 monitor enable 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

29.4.1.2 CMU Frequency Display Register (CMU_FDR)

Figure 280. CMU Frequency Display Register (CMU_FDR)

Address: Base + 0004h

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	FD[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 364. CMU_FDR field descriptions

Field	Description
0:11	Reserved
12:31 FD	Measured frequency bits. This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMNO_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMNO_RMT} \times CMU_MDR[MD]) / CMU_FDR[FD]$

29.4.1.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. [Figure 281](#) and [Table 365](#) shows the CMU_HFREFR register.

Figure 281. CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

Address: Base + 0008h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	HFREF											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Table 365. CMU_HFREFR field descriptions

Field	Description
0:19	Reserved
20:31 HFREF	High Frequency reference value — These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: $(\text{HFREF}/16) \times (f_{\text{CLKMT0_RMN}}/4)$.

29.4.1.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. [Figure 282](#) and [Table 366](#) shows the CMU_HFREFR register.

Figure 282. CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

Address: Base + 000Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	LFREF											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 366. CMU_LFREFR field descriptions

Field	Description
0:19	Reserved
20:31 LFREF	Low Frequency reference value — These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: $(LFREF/16) * (f_{CLKMTO_RMN}/4)$.

29.4.1.5 CMU Interrupt Status Register (CMU_ISR)

Note: : All the flags in CMU_ISR register are set asynchronously. This register must be read only after an interrupt (safe mode entry) is triggered by the CMU as confirmed by MC_RGM. Otherwise, the read access on this register may fetch an incorrect value.

Note: Before entering low-power stop modes, all clock frequency measurements in CMU should be disabled. Failing to do so may result in spurious interrupts.

Figure 283. CMU Interrupt Status Register (CMU_ISR)

Address: Base + 0010h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FHHI	FLLI	OLR ⁽¹⁾
W													— ⁽²⁾	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details.

2. Field is read only.

Table 367. CMU_ISR field descriptions

Field	Description
0:28	Reserved
29 FHHI	CLKMN1 frequency higher than high reference event status—This bit is set by hardware when CLKMN1 frequency becomes higher than HFREF value and CLKMN1 is ‘ON’ as signaled by the MC_ME. It can be cleared by software by writing ‘1’. 0 No FHH event 1 FHH event occurred

Table 367. CMU_ISR field descriptions (continued)

Field	Description
30 FLLI	CLKMN1 frequency less than low reference event status—This bit is set by hardware when CLKMN1 frequency becomes lower than LFREF value and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than $f_{\text{CLKMT0_RMN}}/2^{\text{RCDIV}}$ event status—This bit is set by hardware when the $f_{\text{CLKMN0_RMT}}$ is less than $f_{\text{CLKMT0_RMN}}/2^{\text{RCDIV}}$ frequency and CLKMN0_RMT is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No OLR event 1 OLR event occurred

29.4.1.6 CMU Measurement Duration Register (CMU_MDR)

Figure 284. CMU Measurement Duration Register (CMU_MDR)

Address: Base + 0018h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MD[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 368. CMU_MDR field descriptions

Field	Description
0:11	Reserved
12:31 MD	Measurement duration bits — This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. When CMU_CSR [SFM] = 1, the down-counter starts counting.

29.5 Functional description

This section describes the functionality of the clocks.

29.5.1 Frequency meter

The purpose of frequency meter is to evaluate the deviation from the nominal metered source (such as, CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of metered clocks from a multiplexer that drives a simple Frequency Meter (see [Figure 278](#)). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR as shown in the following equation:

$$\text{Equation 15} \quad f_{sel} = (f_{CLKMN0_RMT} \times MDR[MD]) / FDR[FD]$$

29.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than frequency of CLKMT0_RMN divided by $2^{CMU_CSR[RCDIV]}$ and CLKMN0_RMT is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

Note: f_{CLKMN0_RMT} must be greater than f_{CLKMT0_RMN} divided by 2^{RCDIV} by at least 0.5 MHz in order to guarantee correct f_{CLKMN0_RMT} monitoring.

29.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If f_{CLKMN1} is greater than the reference value determined by fields CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI]
- The CMU asserts the FHH signal

If f_{CLKMN1} is less than a reference value determined by the bits CMU_LFREFR[LFREF] and the CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note: f_{CLKMN1} must be greater than $f_{CLKMT0_RMN} / 4$ by at least 0.5 MHz in order to guarantee correct CLKMN1 monitoring.

30 Clock Generation Module (MC_CGM)

30.1 Introduction

30.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the modules on the chip. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). A set of MC_CGM registers controls the clock dividers which are used for divided system and peripheral clock generation. The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space.

[Figure 285](#) shows the MC_CGM block diagram.

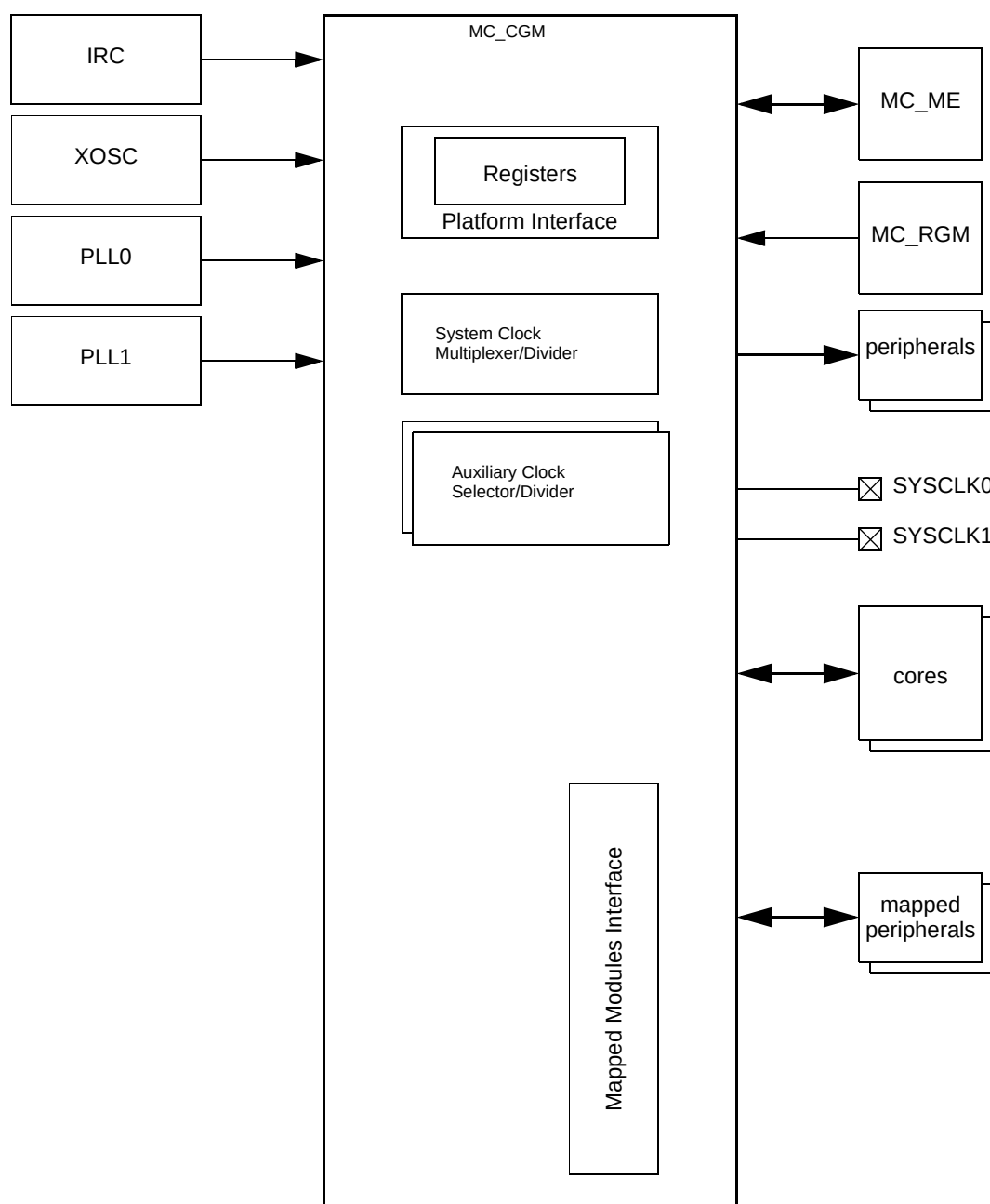


Figure 285. MC_CGM block diagram

30.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control
- performs progressive system clock frequency change depending on MC_ME mode configuration
- contains a set of registers to control clock dividers for divided clock generation
- supports multiple clock sources and maps their address spaces to its memory map
- guarantees glitch-less clock transitions when changing the system clock selection
- supports 8, 16, and 32-bit wide read/write accesses

30.2 External signal description

The MC_CGM delivers output clocks to the SYSCLK0 and SYSCLK1 pins for off-chip use and/or observation.

30.3 Memory map and register definition

Table 369. MC_CGM register description

Address offset	Abbreviation	Name	Size	Access			Location
				User	Supervisor	Test	
0x0680	CGM_AC5_CDC0	Auxiliary Clock 5 Cascaded Divider 0 Configuration Register	word	read	read/write	read/write	on page 940
0x0684	CGM_AC5_CDC1	Auxiliary Clock 5 Cascaded Divider 1 Configuration Register	word	read	read/write	read/write	on page 941
0x0688	CGM_AC5_CDC2	Auxiliary Clock 5 Cascaded Divider 2 Configuration Register	word	read	read/write	read/write	on page 942
0x0690	CGM_AC5_CDC10	Aux Clock 5 Cascaded Divider Configuration 10	word	read	read/write	read/write	on page 942
0x0694	CGM_AC5_CDC11	Aux Clock 5 Cascaded Divider Configuration 11	word	read	read/write	read/write	on page 943
0x0698	CGM_AC5_CDC12	Aux Clock 5 Cascaded Divider Configuration 12	word	read	read/write	read/write	on page 944

Table 369. MC_CGM register description (continued)

Address offset	Abbreviation	Name	Size	Access			Location
				User	Supervisor	Test	
0x069C	CGM_AC5_CDC13	Aux Clock 5 Cascaded Divider Configuration 13	word	read	read/write	read/write	on page 945
0x06A0	CGM_AC5_CDC20	Aux Clock 5 Cascaded Divider Configuration 20	word	read	read/write	read/write	on page 945
0x06A4	CGM_AC5_CDC21	Aux Clock 5 Cascaded Divider Configuration 21	word	read	read/write	read/write	on page 946
0x06A8	CGM_AC5_CDC22	Aux Clock 5 Cascaded Divider Configuration 22	word	read	read/write	read/write	on page 947
0x06AC	CGM_AC5_CDC23	Aux Clock 5 Cascaded Divider Configuration 23	word	read	read/write	read/write	on page 948
0x0700	CGM_PCS_SDUR	PCS Switch Duration Register	byte	read	read/write	read/write	on page 948
0x0704	CGM_PCS_DIVC1	PCS Divider Change Register 1	word	read	read/write	read/write	on page 949
0x0708	CGM_PCS_DIVS1	PCS Divider Start Register 1	word	read	read/write	read/write	on page 949
0x070C	CGM_PCS_DIVE1	PCS Divider End Register 1	word	read	read/write	read/write	on page 950
0x0710	CGM_PCS_DIVC2	PCS Divider Change Register 2	word	read	read/write	read/write	on page 951
0x0714	CGM_PCS_DIVS2	PCS Divider Start Register 2	word	read	read/write	read/write	on page 951
0x0718	CGM_PCS_DIVE2	PCS Divider End Register 2	word	read	read/write	read/write	on page 952
0x0728	CGM_PCS_DIVC4	PCS Divider Change Register 4	word	read	read/write	read/write	on page 953
0x072C	CGM_PCS_DIVS4	PCS Divider Start Register 4	word	read	read/write	read/write	on page 953
0x0730	CGM_PCS_DIVE4	PCS Divider End Register 4	word	read	read/write	read/write	on page 954
0x07E4	CGM_SC_SS	System Clock Select Status Register	word	read	read	read	on page 955
0x07E8	CGM_SC_DC0	System Clock Divider 0 Configuration Register	word	read	read/write	read/write	on page 956

Table 369. MC_CGM register description (continued)

Address offset	Abbreviation	Name	Size	Access			Location
				User	Supervisor	Test	
0x07EC	CGM_SC_DC1	System Clock Divider 1 Configuration Register	word	read	read/write	read/write	on page 957
0x07F0	CGM_SC_DC2	System Clock Divider 2 Configuration Register	word	read	read/write	read/write	on page 958
0x0800	CGM_AC0_SC	Auxiliary Clock 0 Select Control Register	word	read	read/write	read/write	on page 959
0x0804	CGM_AC0_SS	Auxiliary Clock 0 Select Status Register	word	read	read	read	on page 960
0x0808	CGM_AC0_DC0	Auxiliary Clock 0 Divider 0 Configuration Register	word	read	read/write	read/write	on page 960
0x080C	CGM_AC0_DC1	Auxiliary Clock 0 Divider 1 Configuration Register	word	read	read/write	read/write	on page 961
0x0810	CGM_AC0_DC2	Auxiliary Clock 0 Divider 2 Configuration Register	word	read	read/write	read/write	on page 962
0x0814	CGM_AC0_DC3	Auxiliary Clock 0 Divider 3 Configuration Register	word	read	read/write	read/write	on page 962
0x0818	CGM_AC0_DC4	Auxiliary Clock 0 Divider 4 Configuration Register	word	read	read/write	read/write	on page 964
0x0820	CGM_AC1_SC	Auxiliary Clock 1 Select Control Register	word	read	read/write	read/write	on page 964
0x0824	CGM_AC1_SS	Auxiliary Clock 1 Select Status Register	word	read	read	read	on page 965
0x0828	CGM_AC1_DC0	Auxiliary Clock 1 Divider 0 Configuration Register	word	read	read/write	read/write	on page 966

Table 369. MC_CGM register description (continued)

Address offset	Abbreviation	Name	Size	Access			Location
				User	Supervisor	Test	
0x0848	CGM_AC2_DC0	Auxiliary Clock 2 Divider 0 Configuration Register	word	read	read/write	read/write	on page 967
0x084C	CGM_AC2_DC1	Auxiliary Clock 2 Divider 1 Configuration Register	word	read	read/write	read/write	on page 967
0x0860	CGM_AC3_SC	Auxiliary Clock 3 Select Control Register	word	read	read/write	read/write	on page 968
0x0864	CGM_AC3_SS	Auxiliary Clock 3 Select Status Register	word	read	read	read	on page 969
0x0880	CGM_AC4_SC	Auxiliary Clock 4 Select Control Register	word	read	read/write	read/write	on page 970
0x0884	CGM_AC4_SS	Auxiliary Clock 4 Select Status Register	word	read	read	read	on page 971
0x08A8	CGM_AC5_DC0	Auxiliary Clock 5 Divider 0 Configuration Register	word	read	read/write	read/write	on page 972
0x08AC	CGM_AC5_DC1	Auxiliary Clock 5 Divider 1 Configuration Register	word	read	read/write	read/write	on page 973
0x08B0	CGM_AC5_DC2	Auxiliary Clock 5 Divider 2 Configuration Register	word	read	read/write	read/write	on page 974
0x08C0	CGM_AC6_SC	Auxiliary Clock 6 Select Control Register	word	read	read/write	read/write	on page 974
0x08C4	CGM_AC6_SS	Auxiliary Clock 6 Select Status Register	word	read	read	read	on page 976
0x08C8	CGM_AC6_DC0	Auxiliary Clock 6 Divider 0 Configuration Register	word	read	read/write	read/write	on page 976
0x08E0	CGM_AC7_SC	Auxiliary Clock 7 Select Control Register	word	read	read/write	read/write	on page 977

Table 369. MC_CGM register description (continued)

Address offset	Abbreviation	Name	Size	Access			Location
				User	Supervisor	Test	
0x08E4	CGM_AC7_SS	Auxiliary Clock 7 Select Status Register	word	read	read	read	on page 978
0x08E8	CGM_AC7_DC0	Auxiliary Clock 7 Divider 0 Configuration Register	word	read	read/write	read/write	on page 979
0x0900	CGM_AC8_SC	Auxiliary Clock 8 Select Control Register	word	read	read/write	read/write	on page 980
0x0904	CGM_AC8_SS	Auxiliary Clock 8 Select Status Register	word	read	read	read	on page 981
0x0908	CGM_AC8_DC0	Auxiliary Clock 8 Divider 0 Configuration Register	word	read	read/write	read/write	on page 982
0x0920	CGM_AC9_SC	Auxiliary Clock 9 Select Control Register	word	read	read/write	read/write	on page 983
0x0924	CGM_AC9_SS	Auxiliary Clock 9 Select Status Register	word	read	read	read	on page 984
0x0928	CGM_AC9_DC0	Auxiliary Clock 9 Divider 0 Configuration Register	word	read	read/write	read/write	on page 985
0x0940	CGM_AC10_SC	Auxiliary Clock 10 Select Control Register	word	read	read/write	read/write	on page 986
0x0944	CGM_AC10_SS	Auxiliary Clock 10 Select Status Register	word	read	read	read	on page 987
0x0948	CGM_AC10_DC0	Auxiliary Clock 10 Divider 0 Configuration Register	word	read	read/write	read/write	on page 988

Note: Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Table 370. MC_CGM memory map

Address	Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0000 ... 0x003C	IRC registers																
0x0040 ... 0x007C	reserved																
0x0080 ... 0x00BC	XOSC registers																
0x00C0 ... 0x00FC	reserved																
0x0100 ... 0x013C	PLLs registers																
0x0140 ... 0x017C	reserved																
0x0184 ... 0x01BC	reserved																
0x01C4 ... 0x01FC	reserved																
0x0200 ... 0x023C	CMU_A0 registers																
0x0244 ... 0x027C	reserved																
0x0280 ... 0x02BC	reserved																
0x02C0 ... 0x02FC	reserved																
0x0300 ... 0x033C	reserved																

Table 370. MC_CGM memory map (continued)

Address	Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0340 ... 0x037C	reserved																
0x0380 ... 0x03BC	reserved																
0x03C0 ... 0x03FC	reserved																
0x0400 ... 0x043C	reserved																
0x0440 ... 0x047C	reserved																
0x0480 ... 0x04BC	reserved																
0x04C0 ... 0x04FC	reserved																
0x0500 ... 0x053C	reserved																
0x0540 ... 0x057C	reserved																
0x0580 ... 0x05BC	reserved																
0x05C0 ... 0x05FC	reserved																
0x0600 ... 0x063C	reserved																
0x0640 ... 0x067C	reserved																

Table 370. MC_CGM memory map (continued)

Address	Name																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0680	CGM_AC5_CDC0	R	0	0	0	0	0	DIV									
		W															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W															
0x0684	CGM_AC5_CDC1	R	0	0	0	0	0	DIV									
		W															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W															
0x0688	CGM_AC5_CDC2	R	0	0	0	0	0	DIV									
		W															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W															
0x068C	reserved																
0x0690	CGM_AC5_CDC10	R	DE	0	0	0	0	DIV									
		W															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W															
0x0694	CGM_AC5_CDC11	R	DE	0	0	0	0	DIV									
		W															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W															
0x0698	CGM_AC5_CDC12	R	DE	0	0	0	0	DIV									
		W															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W															
0x069C	CGM_AC5_CDC13	R	DE	0	0	0	0	DIV									
		W															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W															
0x06A0	CGM_AC5_CDC20	R	DE	0	0	0	0	DIV									
		W															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W															

Table 370. MC_CGM memory map (continued)

Address	Name	0123456789101112131415																			
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
0x06A4	CGM_AC5_CDC21	R	DE	0	0	0	0	0	DIV												
		W																			
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
		W																			
0x06A8	CGM_AC5_CDC22	R	DE	0	0	0	0	0	DIV												
		W																			
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
		W																			
0x06AC	CGM_AC5_CDC23	R	DE	0	0	0	0	0	DIV												
		W																			
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
		W																			
0x06B0 ... 0x06BC	reserved																				
0x06C0 ... 0x06FC	reserved																				
0x0700	CGM_PCS_SDUR	R	SDUR									0	0	0	0	0	0	0	0		
		W																			
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
		W																			
0x0704	CGM_PCS_DIVC1	R	INIT																		
		W																			
		R	0	0	0	0	0	0	0	0	RATE										
		W																			
0x0708	CGM_PCS_DIVS1	R	0	0	0	0	0	0	0	0	0	0	0	DIVS[19:16]							
		W																			
		R	DIVS[15:0]																		
		W																			
0x070C	CGM_PCS_DIVE1	R	0	0	0	0	0	0	0	0	0	0	0	DIVE[19:16]							
		W																			
		R	DIVE[15:0]																		
		W																			

Table 370. MC_CGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
0x0710	CGM_PCS_DIVC2	R	INIT																	
		W																		
		R	0	0	0	0	0	0	0	0	0	RATE								
		W																		
0x0714	CGM_PCS_DIVS2	R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS[19:16]					
		W																		
		R	DIVS[15:0]																	
		W																		
0x0718	CGM_PCS_DIVE2	R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE[19:16]					
		W																		
		R	DIVE[15:0]																	
		W																		
0x071C ... 0x0724	reserved																			
... 0x0728	CGM_PCS_DIVC4	R	INIT																	
		W																		
		R	0	0	0	0	0	0	0	0	0	RATE								
		W																		
0x072C	CGM_PCS_DIVS4	R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS[19:16]					
		W																		
		R	DIVS[15:0]																	
		W																		
0x0730	CGM_PCS_DIVE4	R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE[19:16]					
		W																		
		R	DIVE[15:0]																	
		W																		
0x0734 ... 0x07CC	reserved																			
0x0734 ... 0x07E0	reserved																			

Table 370. MC_CGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0x07E4	CGM_SC_SS	R	0	0	0	0	SELSTAT					0	0	0	0	SWTRG			SWIP
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																	
0x07E8	CGM_SC_DC0	R	DE	0	0	0	0	0	0	0	0	0	DIV						
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x07EC	CGM_SC_DC1	R	DE	0	0	0	0	0	0	0	0	0	DIV						
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x07F0	CGM_SC_DC2	R	DE	0	0	0	0	0	0	0	0	0	DIV						
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x07F4 ... 0x07FC	reserved																		
0x0800	CGM_AC0_SC	R	0	0	0	0	SELCTL					0	0	0	0	0	0	0	0
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x0804	CGM_AC0_SS	R	0	0	0	0	SELSTAT					0	0	0	0	0	0	0	0
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x0808	CGM_AC0_DC0	R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV				
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	

Table 370. MC_CGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0x080C	CGM_AC0_DC1	R	DE	0	0	0	0	0	0	0	0	DIV							
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x0810	CGM_AC0_DC2	R	DE	0	0	0	0	0	0	0	0	DIV							
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x0814	CGM_AC0_DC3	R	DE	0	0	0	0	0	0	DIV									
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV_FMT		
		W																	
0x0818	CGM_AC0_DC4	R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV				
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																	
0x081C	reserved																		
0x0820	CGM_AC1_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0	
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x0824	CGM_AC1_SS	R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0	
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																	
0x0828	CGM_AC1_DC0	R	DE	0	0	0	0	0	0	0	0	DIV							
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																	
0x082C ... 0x0844	reserved																		

Table 370. MC_CGM memory map (continued)

Address	Name	0123456789101112131415																		
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
0x0848	CGM_AC2_DC0	R	DE	0	0	0	0	0	0	0	0	0	DIV							
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																		
0x084C	CGM_AC2_DC1	R	DE	0	0	0	0	0	0	0	0	0	DIV							
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																		
0x0850 ... 0x085C	reserved																			
0x0860	CGM_AC3_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0	0	
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																		
0x0864	CGM_AC3_SS	R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0	0	
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																		
0x0868 ... 0x087C	reserved																			
0x0880	CGM_AC4_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0	0	
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																		
0x0884	CGM_AC4_SS	R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0	0	
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																		
0x0888 ... 0x08A4	reserved																			

Table 370. MC_CGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
0x08A8	CGM_AC5_DC0	R	DE	0	0	DIV														
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV_FMT			
		W																		
0x08AC	CGM_AC5_DC1	R	DE	0	0	0	0	0	0	0	DIV									
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																		
0x08B0	CGM_AC5_DC2	R	DE	0	0	0	0	0	DIV											
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																		
0x08B4 ... 0x08BC	reserved																			
0x08C0	CGM_AC6_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0		
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																		
0x08C4	CGM_AC6_SS	R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0		
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																		
0x08C8	CGM_AC6_DC0	R	DE	0	0	0	0	0	0	DIV										
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																		
0x08CC ... 0x08DC	reserved																			
0x08E0	CGM_AC7_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0		
		W																		
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																		

Table 370. MC_CGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0x08E4	CGM_AC7_SS	R	0	0	0	0	SELSTAT					0	0	0	0	0	0	0	0
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x08E8	CGM_AC7_DC0	R	DE	0	0	0	0	0	0	DIV									
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x08EC ... 0x08FC	reserved																		
0x0900	CGM_AC8_SC	R	0	0	0	0	SELCTL					0	0	0	0	0	0	0	0
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x0904	CGM_AC8_SS	R	0	0	0	0	SELSTAT					0	0	0	0	0	0	0	0
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x0908	CGM_AC8_DC0	R	DE	0	0	0	0	0	0	0	0	0	DIV						
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x090C ... 0x091C	reserved																		
0x0920	CGM_AC9_SC	R	0	0	0	0	SELCTL					0	0	0	0	0	0	0	0
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x0924	CGM_AC9_SS	R	0	0	0	0	SELSTAT					0	0	0	0	0	0	0	0
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	

Table 370. MC_CGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0928	CGM_AC9_DC0	R	0	0	0	0	0	0	0	0	0	0	DIV					
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0x092C ... 0x093C	reserved																	
0x0940	CGM_AC10_SC	R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0x0944	CGM_AC10_SS	R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0x0948	CGM_AC10_DC0	R	0	0	0	0	0	0	0	0	0	0	0	DIV				
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0x094C ... 0x3FFC	reserved																	

30.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0704, as a half-word at address offset 0x0706, or as a byte at address offset 0x0707.

30.3.1.1 Auxiliary Clock 5 Cascaded Divider 0 Configuration Register (CGM_AC5_CDC0)

This register controls auxiliary clock 5 cascaded divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 286. Auxiliary Clock 5 Cascaded Divider 0 Configuration Register (CGM_AC5_CDC0)

Address 0x0680

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 371. CGM_AC5_CDC0 field descriptions

Field	Description
DE	Divider Enable. 0 Disable auxiliary clock 5 cascaded divider 0 1 Enable auxiliary clock 5 cascaded divider 0
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.2 Auxiliary Clock 5 Cascaded Divider 1 Configuration Register (CGM_AC5_CDC1)

This register controls system clock divider 1.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 287. Auxiliary Clock 5 Cascaded Divider 1 Configuration Register (CGM_AC5_CDC1)

Address 0x0684

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 372. CGM_AC5_CDC1 field descriptions

Field	Description
DE	Divider Enable. 0 Disable auxiliary clock 5 cascaded divider 1 1 Enable auxiliary clock 5 cascaded divider 1
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.3 Auxiliary Clock 5 Cascaded Divider 2 Configuration Register (CGM_AC5_CDC2)

This register controls system clock divider 2.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 288. Auxiliary Clock 5 Cascaded Divider 2 Configuration Register (CGM_AC5_CDC2)

Address 0x0688

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 373. CGM_AC5_CDC2 field descriptions

Field	Description
DE	Divider Enable. 0 Disable auxiliary clock 5 cascaded divider 2 1 Enable auxiliary clock 5 cascaded divider 2
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.4 Auxiliary Clock 5 Cascaded Divider 10 Configuration Register (CGM_AC5_CDC10)

This register controls auxiliary clock 5 cascaded divider 10.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 289. Auxiliary Clock 5 Cascaded Divider 10 Configuration Register (CGM_AC5_CDC10)

Address 0x0690

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 374. CGM_AC5_CDC10 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 10 1 Enable auxiliary clock 5 cascaded divider 10
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_CDC1[DIV] + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.5 Auxiliary Clock 5 Cascaded Divider 11 Configuration Register (CGM_AC5_CDC11)

This register controls auxiliary clock 5 cascaded divider 11.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 290. Auxiliary Clock 5 Cascaded Divider 11 Configuration Register (CGM_AC5_CDC11)

Address 0x0694

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 375. CGM_AC5_CDC11 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 11 1 Enable auxiliary clock 5 cascaded divider 11
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_CDC1[DIV] + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.6 Auxiliary Clock 5 Cascaded Divider 12 Configuration Register (CGM_AC5_CDC12)

This register controls auxiliary clock 5 cascaded divider 12.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Figure 291. Auxiliary Clock 5 Cascaded Divider 12 Configuration Register (CGM_AC5_CDC12)

Address 0x0698

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 376. CGM_AC5_CDC12 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 12 1 Enable auxiliary clock 5 cascaded divider 12
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_CDC1[DIV] + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.7 Auxiliary Clock 5 Cascaded Divider 13 Configuration Register (CGM_AC5_CDC13)

This register controls auxiliary clock 5 cascaded divider 13.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 292. Auxiliary Clock 5 Cascaded Divider 13 Configuration Register (CGM_AC5_CDC13)

Address 0x069C				Access: User read/write, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	DE	0	0	0	0	0	DIV												
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 377. CGM_AC5_CDC13 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 13 1 Enable auxiliary clock 5 cascaded divider 13
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_CDC1[DIV] + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.8 Auxiliary Clock 5 Cascaded Divider 20 Configuration Register (CGM_AC5_CDC20)

This register controls auxiliary clock 5 cascaded divider 20.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 293. Auxiliary Clock 5 Cascaded Divider 20 Configuration Register (CGM_AC5_CDC20)

Address 0x06A0

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 378. CGM_AC5_CDC20 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 20 1 Enable auxiliary clock 5 cascaded divider 20
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_CDC2[DIV] + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.9 Auxiliary Clock 5 Cascaded Divider 21 Configuration Register (CGM_AC5_CDC21)

This register controls auxiliary clock 5 cascaded divider 21.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 294. Auxiliary Clock 5 Cascaded Divider 21 Configuration Register (CGM_AC5_CDC21)

Address 0x06A4

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 379. CGM_AC5_CDC21 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 21 1 Enable auxiliary clock 5 cascaded divider 21
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_CDC2[DIV] + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.10 Auxiliary Clock 5 Cascaded Divider 22 Configuration Register (CGM_AC5_CDC22)

This register controls auxiliary clock 5 cascaded divider 22.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 295. Auxiliary Clock 5 Cascaded Divider 22 Configuration Register (CGM_AC5_CDC22)

Address 0x06A8

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 380. CGM_AC5_CDC22 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 22 1 Enable auxiliary clock 5 cascaded divider 2
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_CDC2[DIV] + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.11 Auxiliary Clock 5 Cascaded Divider 23 Configuration Register (CGM_AC5_CDC23)

This register controls auxiliary clock 5 cascaded divider 23.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 296. Auxiliary Clock 5 Cascaded Divider 23 Configuration Register (CGM_AC5_CDC23)

Address 0x06AC					Access: User read/write, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
R	DE	0	0	0	0	0	DIV													
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 381. CGM_AC5_CDC23 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 23 1 Enable auxiliary clock 5 cascaded divider 23
DIV	Divider Division Value — The resultant peripheral clock will have a period ' $(DIV + 1) \times (CGM_AC5_CDC2[DIV] + 1) \times (CGM_AC5_DC2[DIV] + 1)$ ' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.12 PCS Switch Duration Register (CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See [Section 30.4.1.2, Progressive system clock switching](#), for details on how to set this value. This register is reset only on a power-on reset.

Figure 297. PCS Switch Duration Register (CGM_PCS_SDUR)

Address 0x0700				Access: User read/write, Supervisor read/write, Test read/write							
	0	1	2	3	4	5	6	7			
R	SDUR										
W											
POR	0	0	0	0	0	0	0	0	0	0	

Table 382. CGM_PCS_SDUR field descriptions

Field	Description
SDUR	Switch Duration — This value defines the duration of one PCS clock switch step in terms of 16 MHz int. RC osc. cycles.

30.3.1.13 PCS Divider Change Register 1 (CGM_PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the external crystal osc. on ramp-up and ramp-down, respectively. See [Section 30.4.1.2, Progressive system clock switching](#), for details on how to set these values. This register is reset only on a power-on reset.

Note: *Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 298. PCS Divider Change Register 1 (CGM_PCS_DIVC1)

Address 0x0704					Access: User read/write, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
R	INIT																			
W																				
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1				
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	0	0	0	0	0	0	0	0	RATE											
W																				
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Table 383. CGM_PCS_DIVC1 field descriptions

Field	Description
INIT	Divider Change Initial Value — This is initial change value of the clock divider for the clock ramp-up phase when switching to the external crystal osc..
RATE	Divider Change Rate — This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the external crystal osc..

30.3.1.14 PCS Divider Start Register 1 (CGM_PCS_DIVS1)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the external crystal osc. on ramp-up. Register *n* corresponds to system clock source *n*. See [Section 30.4.1.2, Progressive system clock switching](#), for details on how to set these values. This register is reset only on a power-on reset.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 299. PCS Divider Start Register 1 (CGM_PCS_DIVS1)

Address 0x0708

Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS[19:16]			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS[15:0]															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Table 384. CGM_PCS_DIVS1 field descriptions

Field	Description
DIVS	Divider Start Value — This is the start value of the clock divider for the clock ramp-up phase when switching to the external crystal osc..

30.3.1.15 PCS Divider End Register 1 (CGM_PCS_DIVE1)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the external crystal osc. on ramp-down. See [Section 30.4.1.2, Progressive system clock switching](#), for details on how to set this value. This register is reset only on a power-on reset.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 300. PCS Divider End Register 1 (CGM_PCS_DIVE1)

Address 0x070C

Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE[19:16]			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE[15:0]															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Table 385. CGM_PCS_DIVE1 field descriptions

Field	Description
DIVE	Divider End Value — This is the clock divider end value for the clock ramp-down phase when switching from the external crystal osc..

30.3.1.16 PCS Divider Change Register 2 (CGM_PCS_DIVC2)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL0 PHI on ramp-up and ramp-down, respectively. See [Section 30.4.1.2, Progressive system clock switching](#), for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 301. PCS Divider Change Register 2 (CGM_PCS_DIVC2)

Address 0x0710				Access: User read/write, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	INIT																		
W																			
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1			
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	0	0	0	0	0	0	0	0	RATE										
W																			
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 386. CGM_PCS_DIVC2 field descriptions

Field	Description
INIT	Divider Change Initial Value — This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.
RATE	Divider Change Rate — This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL0 PHI.

30.3.1.17 PCS Divider Start Register 2 (CGM_PCS_DIVS2)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL0 PHI on ramp-up. See [Section 30.4.1.2, Progressive system clock switching](#), for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 302. PCS Divider Start Register 2 (CGM_PCS_DIVS2)

Address 0x0714

Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS[19:16]			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS[15:0]															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Table 387. CGM_PCS_DIVS2 field descriptions

Field	Description
DIVS	Divider Start Value — This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.

30.3.1.18 PCS Divider End Register 2 (CGM_PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL0 PHI on ramp-down. See [Section 30.4.1.2, Progressive system clock switching](#) for details on how to set this value. This register is reset only on a power-on reset.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 303. PCS Divider End Register 2 (CGM_PCS_DIVE2)

Address 0x0718

Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE[19:16]			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE[15:0]															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Table 388. CGM_PCS_DIVE2 field descriptions

Field	Description
DIVE	Divider End Value — This is the clock divider end value for the clock ramp-down phase when switching from the PLL0 PHI.

30.3.1.19 PCS Divider Change Register 4 (CGM_PCS_DIVC4)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL1 PHI on ramp-up and ramp-down, respectively. See [Section 30.4.1.2, Progressive system clock switching](#), for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 304. PCS Divider Change Register 4 (CGM_PCS_DIVC4)

Address 0x0728

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INIT															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 389. CGM_PCS_DIVS2 field descriptions

Field	Description
INIT	Divider Change Initial Value — This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.
RATE	Divider Change Rate — This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL1 PHI.

30.3.1.20 PCS Divider Start Register 4 (CGM_PCS_DIVS4)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL1 PHI on ramp-up. See [Section 30.4.1.2, Progressive system clock switching](#) for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 305. PCS Divider Start Register 4 (CGM_PCS_DIVS4)

Address 0x072C

Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS[19:16]			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS[15:0]															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Table 390. CGM_PCS_DIVS4 field descriptions

Field	Description
DIVS	Divider Start Value — This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.

30.3.1.21 PCS Divider End Register 4 (CGM_PCS_DIVE4)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL1 PHI on ramp-down. See [Section 30.4.1.2, Progressive system clock switching](#), for details on how to set this value. This register is reset only on a power-on reset.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 306. PCS Divider End Register 4 (CGM_PCS_DIVE4)

Address 0x0730

Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE[19:16]			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE[15:0]															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Table 391. CGM_PCS_DIVE4 field descriptions

Field	Description
DIVE	Divider End Value — This is the clock divider end value for the clock ramp-down phase when switching from the PLL1 PHI.

30.3.1.22 System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Figure 307. System Clock Select Status Register (CGM_SC_SS)

Address 0x07E4

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	SWTRG			SWIP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 392. CGM_SC_SS field descriptions

Field	Description
SELSTAT	System Clock Source Selection Status — This value indicates the current source for the system clock. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled
SWTRG	Switch Trigger cause — This value indicates the cause for the latest clock source switch. 000 reserved 001 switch after request from MC_ME succeeded 010 switch after request from MC_ME failed due inactive target clock 011 switch after request from MC_ME failed due inactive current clock 100 switch to 16 MHz int. RC osc. due to SAFE mode request or reset succeeded 101 switch to 16 MHz int. RC osc. due to SAFE mode request or reset, but current clock source was inactive 110 reserved 111 reserved
SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress

30.3.1.23 System Clock Divider 0 Configuration Register (CGM_SC_DC0)

This register controls system clock divider 0. See [Section 30.4.1.4.1, System clock divider synchronization](#), for details on division value limitations.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 308. System Clock Divider 0 Configuration Register (CGM_SC_DC0)

Address 0x07E8

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 393. CGM_SC_DC0 field descriptions

Field	Description
DE	Divider 0 Enable— This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
DIV	Divider 1 Division Value — The resultant fast XBAR clock will have a period 'DIV + 1' times that of the system clock. If DE is set to '0' (divider 1 is disabled), any write access to the DIV field is ignored and the fast XBAR clock remains disabled.

30.3.1.24 System Clock Divider 1 Configuration Register (CGM_SC_DC1)

This register controls system clock divider 1. See [Section 30.4.1.4.1, System clock divider synchronization](#), for details on division value limitations.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 309. System Clock Divider 1 Configuration Register (CGM_SC_DC1)

Address 0x07EC

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 394. CGM_SC_DC1 field descriptions

Field	Description
DE	Divider 1 Enable—This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
DIV	Divider 1 Division Value — The resultant slow XBAR clock will have a period 'DIV + 1' times that of the system clock. If DE is set to '0' (divider 1 is disabled), any write access to the DIV field is ignored and the slow XBAR clock remains disabled.

30.3.1.25 System Clock Divider 2 Configuration Register (CGM_SC_DC2)

This register controls system clock divider 2. See [Section 30.4.1.4.1, System clock divider synchronization](#), for details on division value limitations.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 310. System Clock Divider 2 Configuration Register (CGM_SC_DC2)

Address 0x07F0				Access: User read/write, Supervisor read/write, Test read/write												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 395. CGM_SC_DC2 field descriptions

Field	Description
DE	Divider 2 Enable — This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
DIV	Divider 2 Division Value — The resultant PBRIDGE clock will have a period 'DIV + 1' times that of the system clock. If DE is set to '0' (divider 2 is disabled), any write access to the DIV field is ignored and the PBRIDGE clock remains disabled.

30.3.1.26 Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 0 divider 0: peripheral clock
- divided by auxiliary clock 0 divider 1: sigma delta ADC clock
- divided by auxiliary clock 0 divider 2: SAR ADC clock
- divided by auxiliary clock 0 divider 3: DSPI clock 0
- divided by auxiliary clock 0 divider 4: DSPI clock 1

See [Figure 348](#) for details.

Figure 311. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

Address 0x0800				Access: User read/write, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 396. CGM_AC0_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 0 Source Selection Control — Selects the source for auxiliary clock 0. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

30.3.1.27 Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.

Figure 312. Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

Address 0x0804								Access: User read, Supervisor read, Test read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 397. CGM_AC0_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 0 Source Selection Status — This value indicates the current source for auxiliary clock 0. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

30.3.1.28 Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

This register controls auxiliary clock 0 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 313. Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

Address 0x0808

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 398. CGM_AC0_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 0 divider 0 1 Enable auxiliary clock 0 divider 0
DIV	Divider Division Value — The resultant peripheral clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

30.3.1.29 Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

This register controls auxiliary clock 0 divider 1.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 314. Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

Address 0x080C

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 399. CGM_AC0_DC1 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 0 divider 1 1 Enable auxiliary clock 0 divider 1
DIV	Divider Division Value — The resultant sigma delta ADC clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the sigma delta ADC clock remains disabled.

30.3.1.30 Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

This register controls auxiliary clock 0 divider 2.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 315. Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

Address 0x080C				Access: User read/write, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	DE	0	0	0	0	0	0	0	0	DIV									
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 400. CGM_AC0_DC2 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 0 divider 2 1 Enable auxiliary clock 0 divider 2
DIV	Divider Division Value — The resultant SAR ADC clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the SAR ADC clock remains disabled.

30.3.1.31 Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)

This register controls auxiliary clock 0 divider 3. See [Section 30.4.3.2, Fractional divider details](#), for divider characteristics.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 316. Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)

Address 0x0814

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV_FMT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 401. CGM_AC0_DC3 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 0 divider 3 1 Enable auxiliary clock 0 divider 3
DIV	Divider Division Value — The resultant DSPI clock 0 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the DSPI clock 0 remains disabled.
DIV_FMT	Divider Value Format — This value indicates the format of the division value for calculating the division factor 00 Division factor = (DIV + 1) / 1 01 Division factor = (DIV + 1) / 10 10 Division factor = (DIV + 1) / 100 11 Division factor = (DIV + 1) / 1,000

30.3.1.32 Auxiliary Clock 0 Divider 4 Configuration Register (CGM_AC0_DC4)

This register controls auxiliary clock 0 divider 4.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 317. Auxiliary Clock 0 Divider 4 Configuration Register (CGM_AC0_DC4)

Address 0x0818

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 402. CGM_AC0_DC4 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 0 divider 4 1 Enable auxiliary clock 0 divider 4
DIV	Divider Division Value — The resultant DSPI clock 1 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the SAR ADC clock remains disabled.

30.3.1.33 Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 1 divider 0: LFAST clock

See [Figure 349](#) for details.

Figure 318. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

Address 0x0820

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 403. CGM_AC1_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 1 Source Selection Control — This value selects the current source for auxiliary clock 1.
	0000 reserved
	0001 external crystal osc.
	0010 PLL0 PHI
	0011 reserved
	0100 reserved
	0101 LFAST-SysClk pin
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.34 Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

This register provides the current auxiliary clock 0 source selection.

Figure 319. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

Address 0x0824

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 404. CGM_AC1_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 1 Source Selection Status — This value indicates the current source for auxiliary clock 1.
	0000 reserved
	0001 external crystal osc.
	0010 PLL0 PHI
	0011 reserved
	0100 reserved
	0101 LFAST-SysClk pin
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.35 Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

This register controls auxiliary clock 0 divider 0.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 320. Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

Address 0x0828

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 405. CGM_AC1_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 1 divider 0 1 Enable auxiliary clock 1 divider 0
DIV	Divider Division Value — The resultant LFAST clock will have a period 'DIV + 1' times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the LFAST clock remains disabled.

30.3.1.36 Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)

This register controls auxiliary clock 2 divider 0 when the “CAN jitter” feature is not enabled (see [Section 30.4.3.1, Divider jitter injection](#)).

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 321. Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)

Address 0x0848				Access: User read/write, Supervisor read/write, Test read/write												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 406. CGM_AC2_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 2 divider 0 1 Enable auxiliary clock 2 divider 0
DIV	Divider Division Value — The resultant FlexRay clock will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the FlexRay clock remains disabled.

30.3.1.37 Auxiliary Clock 2 Divider 1 Configuration Register (CGM_AC2_DC1)

This register controls auxiliary clock 2 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 322. Auxiliary Clock 2 Divider 1 Configuration Register (CGM_AC2_DC1)

Address 0x084C

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 407. CGM_AC2_DC1 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 2 divider 1 1 Enable auxiliary clock 2 divider 1
DIV	Divider Division Value — The resultant SENT clock will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the SENT clock remains disabled.

30.3.1.38 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

This register is used to select the current clock source for the PLL0 reference clock.

See [Figure 351](#) for details.

Figure 323. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

Address 0x0860

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 408. CGM_AC3_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 3 Source Selection Control — This value selects the current source for auxiliary clock 3.
	0000 16 MHz int. RC osc.
	0001 external crystal osc.
	0010 reserved
	0011 reserved
	0100 reserved
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.39 Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

Figure 324. Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

Address 0x0864

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 409. CGM_AC3_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 3 Source Selection Status — This value indicates the current source for auxiliary clock 3.
	0000 16 MHz int. RC osc.
	0001 external crystal osc.
	0010 reserved
	0011 reserved
	0100 reserved
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.40 Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

This register is used to select the current clock source for the PLL1 reference clock.

See [Figure 352](#) for details.

Figure 325. Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

Address 0x0880

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 410. CGM_AC4_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 4 Source Selection Control — This value selects the current source for auxiliary clock 4.
	0000 reserved
	0001 external crystal osc.
	0010 reserved
	0011 PLL0 PHI1
	0100 reserved
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.41 Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

This register provides the current auxiliary clock 4 source selection.

Figure 326. Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

Address 0x0884

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 411. CGM_AC4_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 4 Source Selection Status — This value indicates the current source for auxiliary clock 4.
	0000 reserved
	0001 external crystal osc.
	0010 reserved
	0011 PLL0 PHI1
	0100 reserved
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.42 Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)

This register controls auxiliary clock 5 divider 0. See [Section 30.4.3.2, Fractional divider details](#), for divider characteristics.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 327. Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)

Address 0x08A8

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	DIV												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV_FMT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 412. CGM_AC5_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 divider 0 1 Enable auxiliary clock 5 divider 0
DIV	Divider Division Value — The resultant PSI5_f189 clock will have a period 'DIV + 1' times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the PSI5_f189 clock remains disabled.
DIV_FMT	Divider Value Format — This value indicates the format of the division value for calculating the division factor 00 Division factor = (DIV + 1) / 1 01 Division factor = (DIV + 1) / 10 10 Division factor = (DIV + 1) / 100 11 Division factor = (DIV + 1) / 1,000

30.3.1.43 Auxiliary Clock 5 Divider 1 Configuration Register (CGM_AC5_DC1)

This register controls auxiliary clock 5 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 328. Auxiliary Clock 5 Divider 1 Configuration Register (CGM_AC5_DC1)

Address 0x08AC

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 413. CGM_AC5_DC1 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 divider 1 1 Enable auxiliary clock 5 divider 1
DIV	Divider Division Value — The resultant PSI5_f125 clock will have a period 'DIV + 1' times that of auxiliary clock 5. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the PSI5_f125 clock remains disabled.

30.3.1.44 Auxiliary Clock 5 Divider 2 Configuration Register (CGM_AC5_DC2)

This register controls auxiliary clock 5 divider 2.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 329. Auxiliary Clock 5 Divider 2 Configuration Register (CGM_AC5_DC2)

Address 0x08B0								Access: User read/write, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15							
R	DE	0	0	0	0	0	DIV																
W																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 414. CGM_AC5_DC2 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 5 divider 2 1 Enable auxiliary clock 5 divider 2
DIV	Divider Division Value — The resultant PSI5_1us clock will have a period 'DIV + 1' times that of auxiliary clock 5. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the PSI5_1us clock remains disabled.

30.3.1.45 Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 6 divider 0: SYSCLK0 pin clock

See [Figure 354](#) for details.

Figure 330. Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)

Address 0x08C0				Access: User read/write, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 415. CGM_AC6_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 6 Source Selection Control — This value selects the current source for auxiliary clock 6.
	0000 16 MHz int. RC osc.
	0001 external crystal osc.
	0010 PLL0 PHI
	0011 reserved
	0100 PLL1 PHI
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.46 Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)

This register provides the current auxiliary clock 6 source selection.

Figure 331. Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)

Address 0x08C4								Access: User read, Supervisor read, Test read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 416. CGM_AC6_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 6 Source Selection Status — This value indicates the current source for auxiliary clock 6. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

30.3.1.47 Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)

This register controls auxiliary clock 6 divider 0.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 332. Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)

Address 0x08A8

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 417. CGM_AC6_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 6 divider 0 1 Enable auxiliary clock 6 divider 0
DIV	Divider Division Value — The resultant SYSCLK0 pin clock will have a period 'DIV + 1' times that of auxiliary clock 6. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SYSCLK0 pin clock remains disabled.

30.3.1.48 Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 7 divider 0: SYSCLK1 pin clock

See [Figure 355](#) for details.

Figure 333. Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)

Address 0x08E0

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 418. CGM_AC7_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 7 Source Selection Control — This value selects the current source for auxiliary clock 7.
	0000 16 MHz int. RC osc.
	0001 external crystal osc.
	0010 PLL0 PHI
	0011 reserved
	0100 PLL1 PHI
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.49 Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)

This register provides the current auxiliary clock 7 source selection.

Figure 334. Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)

Address 0x08E0

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 419. CGM_AC7_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 7 Source Selection Status — This value indicates the current source for auxiliary clock 7.
	0000 16 MHz int. RC osc.
	0001 external crystal osc.
	0010 PLL0 PHI
	0011 reserved
	0100 PLL1 PHI
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.50 Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)

This register controls auxiliary clock 7 divider 0.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 335. Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)

Address 0x08E8

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 420. CGM_AC7_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 7 divider 0 1 Enable auxiliary clock 7 divider 0
DIV	Divider Division Value — The resultant SYSCLK1 pin clock will have a period 'DIV + 1' times that of auxiliary clock 7. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SYSCLK1 pin clock remains disabled.

30.3.1.51 Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 8 divider 0: CAN clock

See [Figure 356](#) for details.

Figure 336. Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

Address 0x0900

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 421. CGM_AC8_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 8 Source Selection Control — This value selects the current source for auxiliary clock 8.
	0000 reserved
	0001 external crystal osc.
	0010 PLL0 PHI
	0011 reserved
	0100 reserved
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.52 Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

This register provides the current auxiliary clock 8 source selection.

Figure 337. Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

Address 0x0904

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 422. CGM_AC8_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 8 Source Selection Status — This value indicates the current source for auxiliary clock 8.
	0000 reserved
	0001 external crystal osc.
	0010 PLL0 PHI
	0011 reserved
	0100 reserved
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.53 Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

This register controls auxiliary clock 8 divider 0 when the “CAN jitter” feature is not enabled (see [Section 30.4.3.1, Divider jitter injection](#)).

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Figure 338. Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

Address 0x0908

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 423. CGM_AC8_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 8 divider 0 1 Enable auxiliary clock 8 divider 0
DIV	Divider Division Value — The resultant CAN clock will have a period 'DIV + 1' times that of auxiliary clock 8. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CAN clock remains disabled.

30.3.1.54 Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 9 divider 0: RTI clock

See [Figure 357](#) for details.

Figure 339. Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)

Address 0x0920

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 424. Auxiliary Clock 9 Select Control Register (CGM_AC9_SC) field descriptions

Field	Description
SELCTL	Auxiliary Clock 9 Source Selection Control — This value selects the current source for auxiliary clock 9.
	0000 16 MHz int. RC osc.
	0001 external crystal osc.
	0010 reserved
	0011 reserved
	0100 reserved
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.55 Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)

This register provides the current auxiliary clock 9 source selection.

Figure 340. Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)

Address 0x0924

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 425. CGM_AC9_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 9 Source Selection Status — This value indicates the current source for auxiliary clock 9.
	0000 16 MHz int. RC osc.
	0001 external crystal osc.
	0010 reserved
	0011 reserved
	0100 reserved
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.56 Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)

This register controls auxiliary clock 9 divider 0.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.*

Figure 341. Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)

Address 0x0928

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 426. CGM_AC9_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 9 divider 0 1 Enable auxiliary clock 9 divider 0
DIV	Divider Division Value — The resultant RTI clock will have a period 'DIV + 1' times that of auxiliary clock 9. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the RTI clock remains disabled.

30.3.1.57 Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)

This register is used to select the current clock source for the Ethernet TXCLK_REF clock.

See [Figure 358](#) for details.

Figure 342. Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)

Address 0x0940

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 427. CGM_AC10_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 10 Source Selection Control — This value selects the current source for auxiliary clock 10.
	0000 reserved
	0001 reserved
	0010 reserved
	0011 reserved
	0100 reserved
	0101 reserved
	0110 REF_CLK pin
	0111 TXCLK pin
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.58 Auxiliary Clock 10 Select Status Register (CGM_AC10_SS)

This register provides the current auxiliary clock 10 source selection.

Figure 343. Auxiliary Clock 10 Select Status Register (CGM_AC10_SS)

Address 0x0944

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 428. CGM_AC10_SS field descriptions

Field	Description
SELSTAT	Auxiliary Clock 10 Source Selection Status — This value indicates the current source for auxiliary clock 10.
	0000 reserved
	0001 reserved
	0010 reserved
	0011 reserved
	0100 reserved
	0101 reserved
	0110 REF_CLK pin
	0111 TXCLK pin
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 reserved

30.3.1.59 Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)

This register controls auxiliary clock 10 divider 0.

Note: *Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.*

Figure 344. Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)

Address 0x0948

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 429. CGM_AC10_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 10 divider 0 1 Enable auxiliary clock 10 divider 0
DIV	Divider Division Value — The resultant Ethernet REF clock will have a period 'DIV + 1' times that of auxiliary clock 10. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the Ethernet REF clock remains disabled.

30.4 Functional description

30.4.1 System clock generation

[Figure 345](#) shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the 16 MHz int. RC osc. as the system clock and to ignore the system clock select.

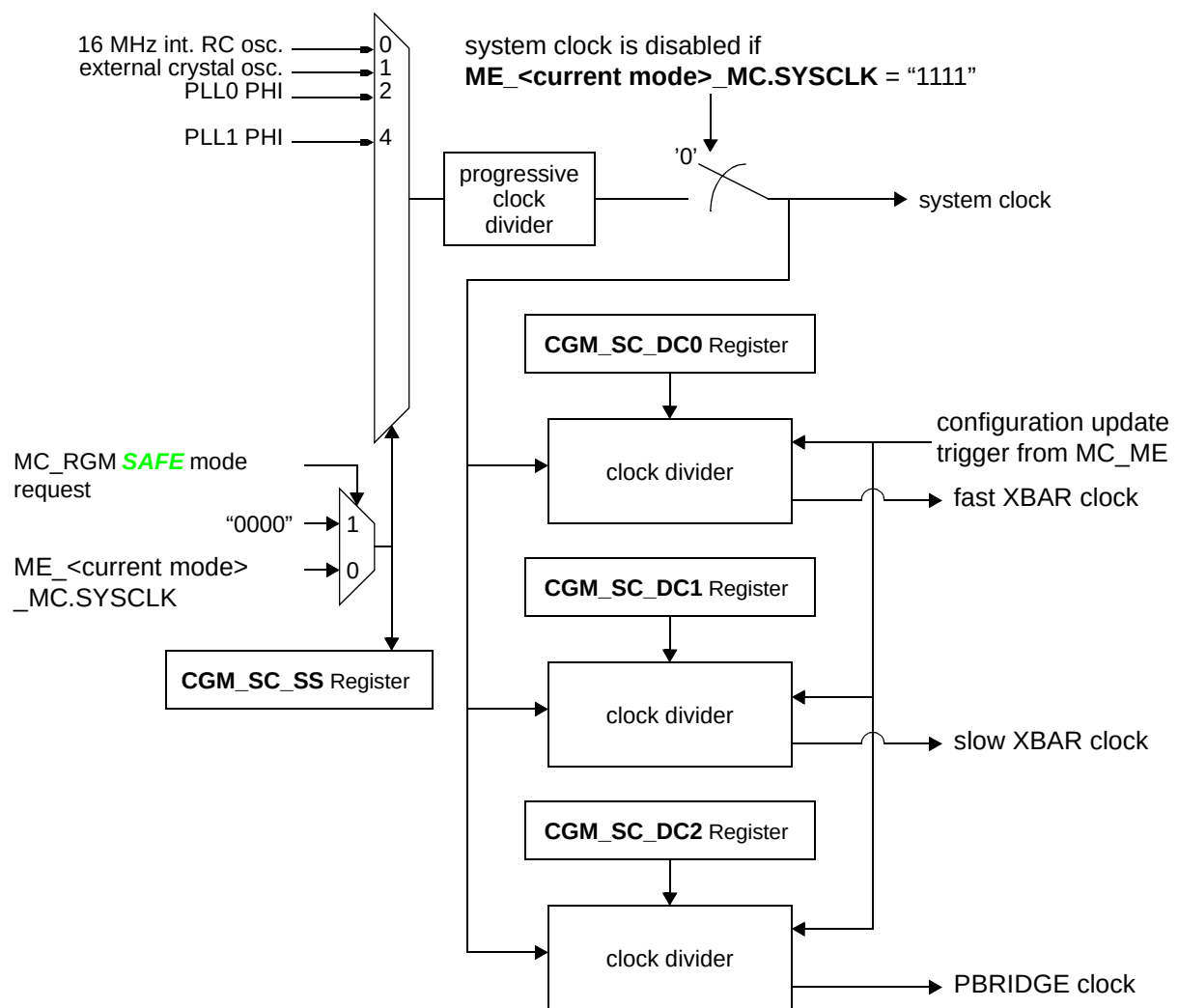


Figure 345. MC_CGM system clock generation overview

30.4.1.1 System clock source selection

During normal operation, the system clock selection is controlled

- on a SAFE mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

30.4.1.2 Progressive system clock switching

In order to prevent sudden voltage drops and overshoots due to frequency and load changes, the MC_ME requests the MC_CGM to ramp the system clock frequency down and/or up based on the power level values of the current and target modes. During ramp-down, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVEN registers, where n corresponds to the current system clock source selection. During ramp-up, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVSn registers, where n corresponds to the target system clock source selection.

30.4.1.2.1 Generic clock change requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in [Equation 16](#).

$$\text{Equation 16} \quad a_{max} = \frac{f_{chg}}{f_{src}}$$

30.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field CGM_PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of 16 MHz int. RC osc.. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

30.4.1.2.3 Configuration of CGM_PCS_DIVCn[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the PCS_DIVCn[RATE] register is given in [Table 430](#).

Table 430. MC_CGM CGM_PCS_DIVCn[RATE] values

a_{max}	d	CGM_PCS_DIVCn[RATE]
0.05	0.012	12
0.10	0.048	48
0.15	0.112	112
0.20	0.184	184

30.4.1.2.4 Generic clock change properties

The number k of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in [Equation 17](#).

$$\text{Equation 17} \quad k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2 \left(1 - \frac{f_{src}}{16\text{MHz}} \right)}{d}} \right\rceil$$

30.4.1.2.5 Clock ramp-down

The clock ramp-down starts with the divider value 1 and with the given divider increment value PCS_DIVCn[RATE] and ends with the given divider value PCS_DIVEn[DIVE].

The divider end value for clock ramp-down is given in [Equation 18](#).

$$\text{Equation 18} \quad \text{PCS_DIVEn[DIVE]} = \frac{f_{src}}{16\text{MHz}} \times 1000 - 1$$

Where f_{curr} is the frequency of the currently selected system clock source.

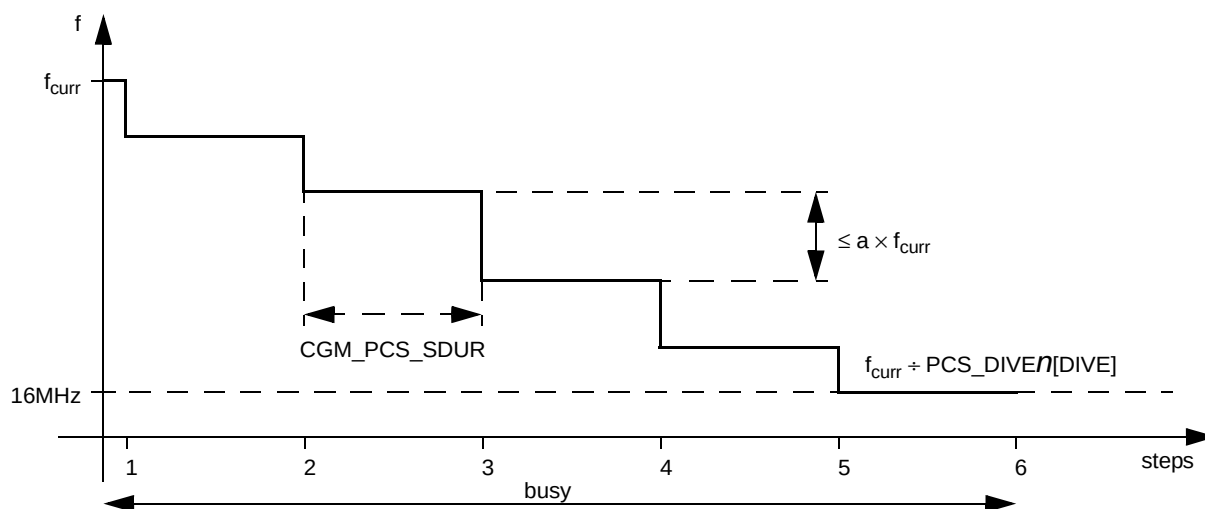


Figure 346. MC_CGM System Clock Ramp-Down Timing (k = 6 example)

30.4.1.2.6 Clock ramp-up

The clock ramp-up starts with the given divider value PCS_DIVSn[DIVS] and with the given divider decrement value PCD_DIVCn[INIT] and ends with the divider value 1.

The initial divider change start value PCS_DIVCn[INIT] for system clock ramp-up is given in [Equation 19](#).

Equation 19 $PCS_DIVCn[INIT] = d \times k \times 1000 - 1$

Where k is calculated from [Equation 17](#) using the target system clock source frequency f_{targ} . The divider start value for clock ramp-up is given in [Equation 20](#).

Equation 20 $PCS_DIVSn[DIVS] = \left(1 + d \frac{k(k+1)}{2}\right) \times 1000 - 1$

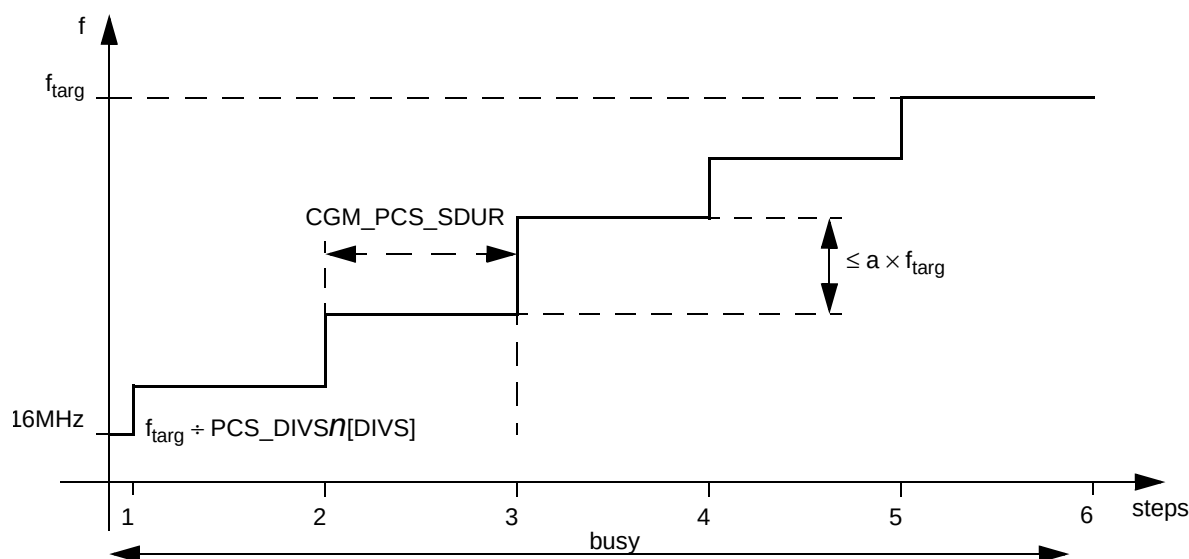


Figure 347. MC_CGM System Clock Ramp-Up Timing (k = 6 example)

30.4.1.3 System clock disable

During the TEST mode, the system clock can be disabled by the MC_ME.

30.4.1.4 System clock dividers

The MC_CGM generates the following derived clocks from the system clock:

- fast XBAR clock - controlled by the CGM_SC_DC0 register
- slow XBAR clock - controlled by the CGM_SC_DC1 register
- PBRIDGE clock - controlled by the CGM_SC_DC2 register

30.4.1.4.1 System clock divider synchronization

The system clock dividers are synchronized to each other such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This, however, imposes limitations on the division factors which can be used. In order for the synchronization to work properly, each division factor must be selected such that its value is an integer multiple of each division factor that has a lower value.

[Table 431](#) shows an example of a correct divider configuration.

Example 4 Correct system clock divider configuration

Table 431. MC_CGM Example System Clock Division Values Compatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	CGM_SC_DC0[DIV] = 0
1	2	CGM_SC_DC1[DIV] = 1
2	6	CGM_SC_DC2[DIV] = 5

In this case, system clock divider 1 has a division factor greater than that of system clock divider 0. Since 2 is an integer multiple of 1, its configuration is correct. Also, system clock divider 3 has a division factor that is greater than those of both system clock dividers 0 and 1. Since 6 is an integer multiple of 1 and 6 is an integer multiple of 2, its configuration is also correct.

[Table 432](#) shows an example of an incorrect divider configuration.

Example 5 Incorrect system clock divider configuration

Table 432. MC_CGM Example System Clock Division Values Compatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	CGM_SC_DC0[DIV] = 0
1	4	CGM_SC_DC1[DIV] = 3
2	6	CGM_SC_DC2[DIV] = 5

In this case, system clock divider 1 has a division factor greater than that of system clock divider 0. Since 4 is an integer multiple of 1, its configuration is correct. Also, system clock divider 3 has a division factor that is greater than those of both system clock dividers 0 and 1. Since 6 is not an integer multiple of 4, its configuration is incorrect.

Caution: Configuring the system clock dividers incorrectly will cause the divided clocks to become unsynchronized. This may lead to lost communication between the clock domains and/or the inability to further change the system clock divider division factors without first disabling them or performing a reset.

30.4.1.4.2 System clock divider update triggering

As opposed to the auxiliary clock dividers, which update immediately on a configuration change, the system clock dividers are updated only with the next software triggered mode change. This is to ensure that the system clock dividers are updated simultaneously and that no intermediate inconsistent configurations can occur. It also ensures that the divider configurations are aligned with the target mode's system clock selection.

The system clock divider configuration update sequence is as follows:

1. Configure the target mode in the MC_ME to select the desired system clock source.
2. Configure the CGM_SC_DC0...2 registers as desired and aligned with the target system clock source frequency making sure that the dividers are configured consistently as described in [Section 30.4.1.4.1, System clock divider synchronization](#).
3. Request a mode change in the MC_ME to the desired target mode.
4. When mode change has completed and the system clock switching has been successful, the new system clock divider configuration will be in effect.

See [Chapter 62: Mode Entry Module \(MC_ME\)](#), for details on how to configure modes and make mode change requests.

30.4.2 Auxiliary clock generation

Figure 348 through *Figure 358* show the block diagrams of the generation logic for the various auxiliary clocks. See the following sections for auxiliary clock selection control:

- [Section 30.3.1.26, Auxiliary Clock 0 Select Control Register \(CGM_AC0_SC\)](#)
- [Section 30.3.1.33, Auxiliary Clock 1 Select Control Register \(CGM_AC1_SC\)](#)
- [Section 30.3.1.38, Auxiliary Clock 3 Select Control Register \(CGM_AC3_SC\)](#)
- [Section 30.3.1.40, Auxiliary Clock 4 Select Control Register \(CGM_AC4_SC\)](#)
- [Section 30.3.1.45, Auxiliary Clock 6 Select Control Register \(CGM_AC6_SC\)](#)
- [Section 30.3.1.48, Auxiliary Clock 7 Select Control Register \(CGM_AC7_SC\)](#)
- [Section 30.3.1.51, Auxiliary Clock 8 Select Control Register \(CGM_AC8_SC\)](#)
- [Section 30.3.1.54, Auxiliary Clock 9 Select Control Register \(CGM_AC9_SC\)](#)
- [Section 30.3.1.57, Auxiliary Clock 10 Select Control Register \(CGM_AC10_SC\)](#)

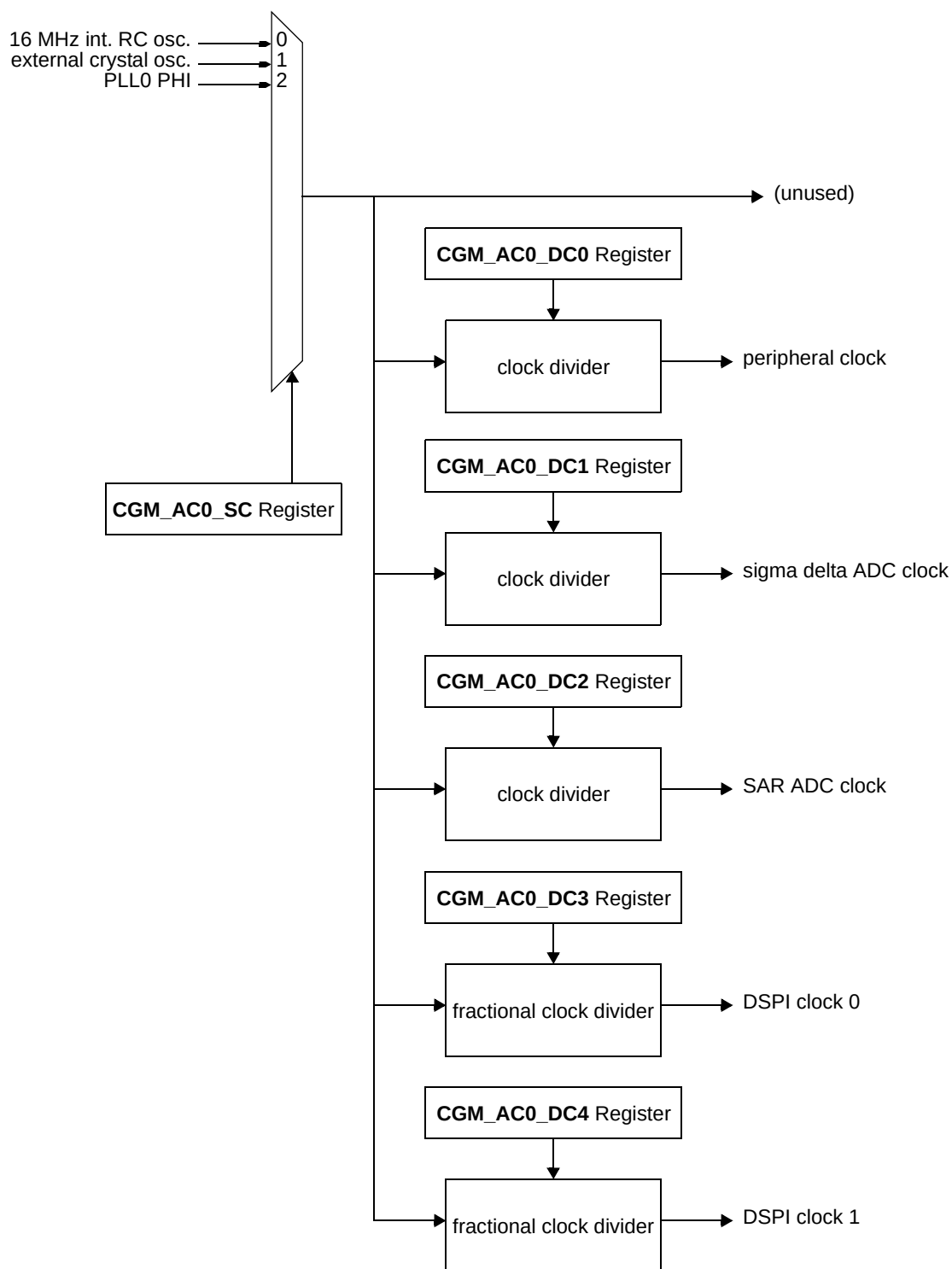


Figure 348. MC_CGM auxiliary clock 0 generation overview

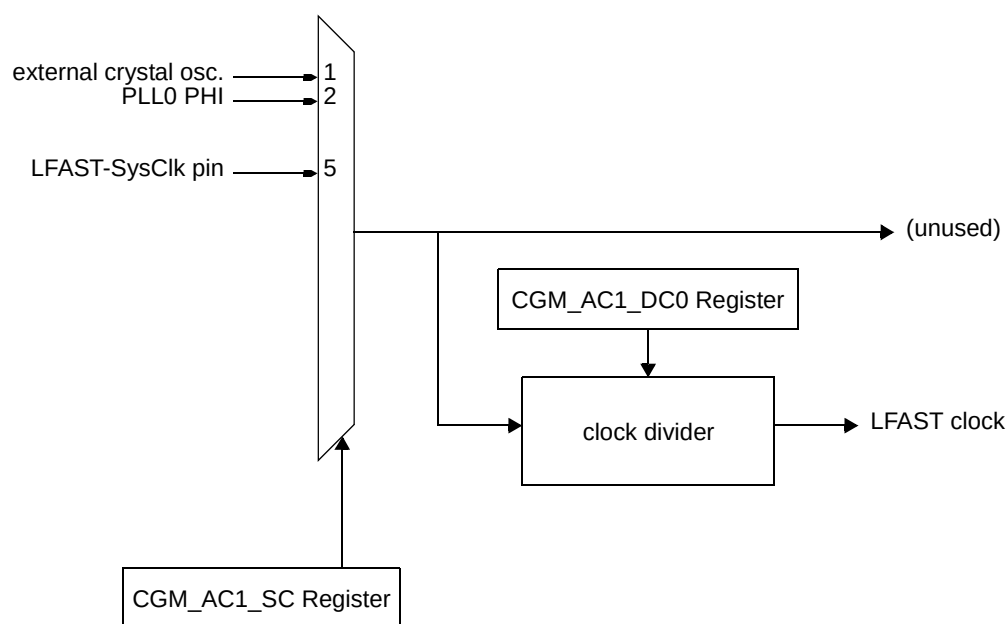


Figure 349. MC_CGM auxiliary clock 1 generation overview

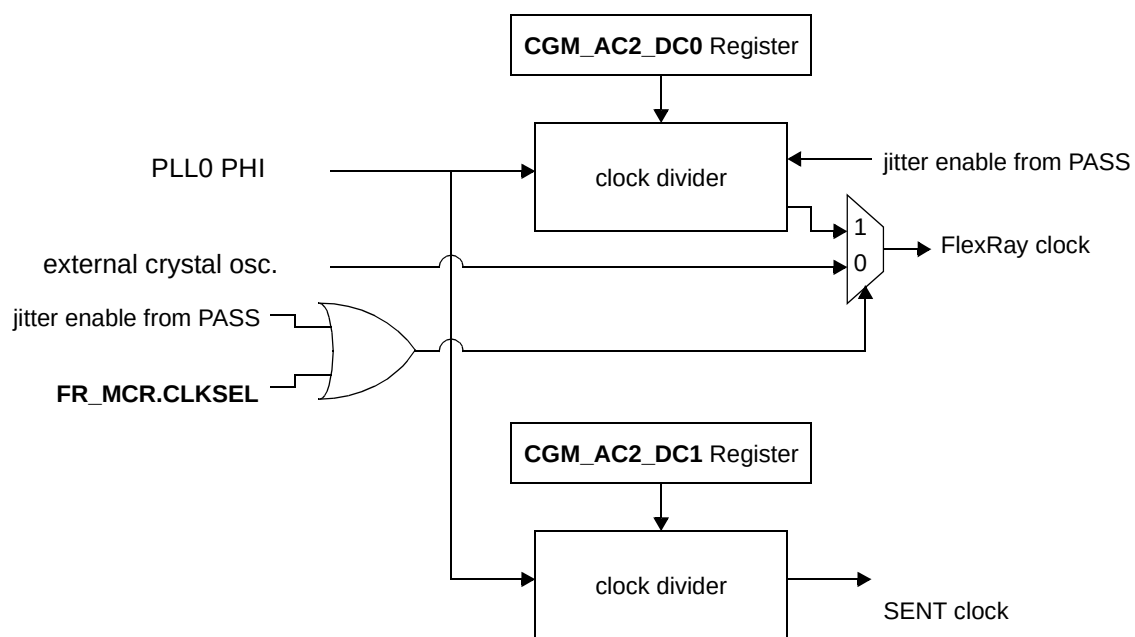


Figure 350. MC_CGM auxiliary clock 2 generation overview

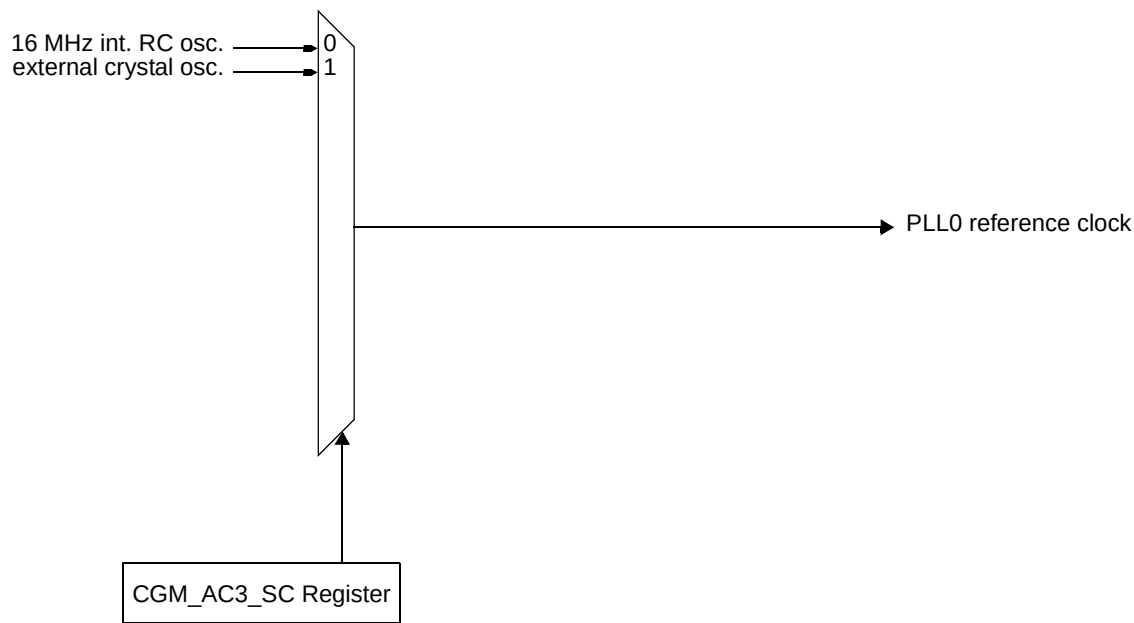


Figure 351. MC_CGM auxiliary clock 3 generation overview

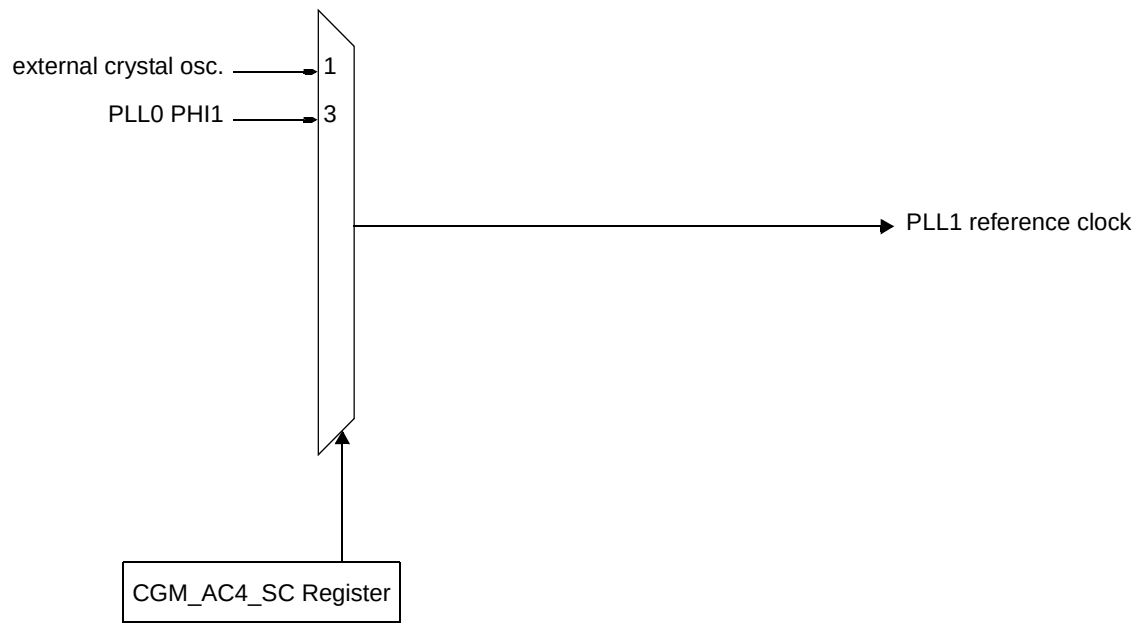


Figure 352. MC_CGM auxiliary clock 4 generation overview

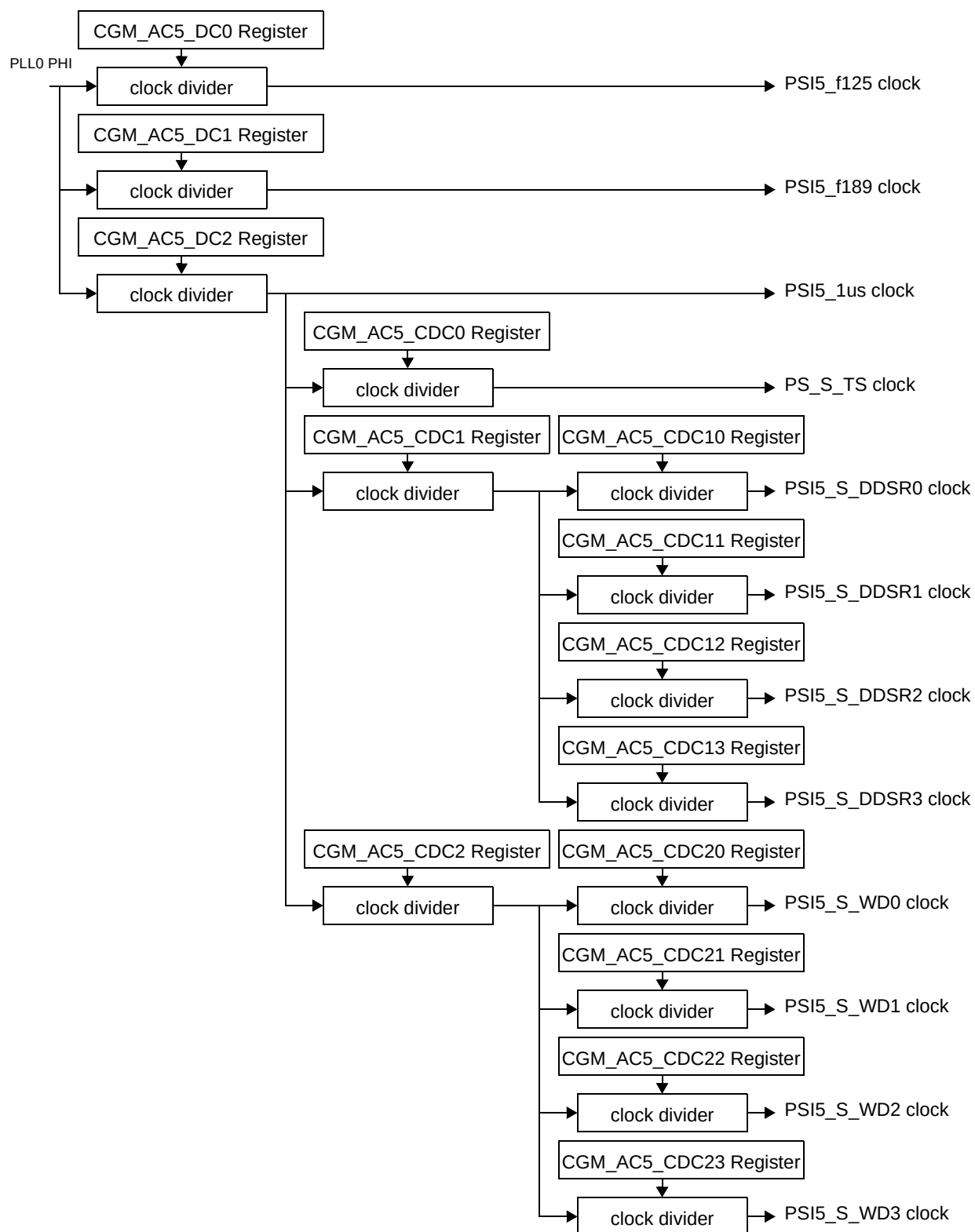


Figure 353. MC_CGM auxiliary clock 5 generation overview

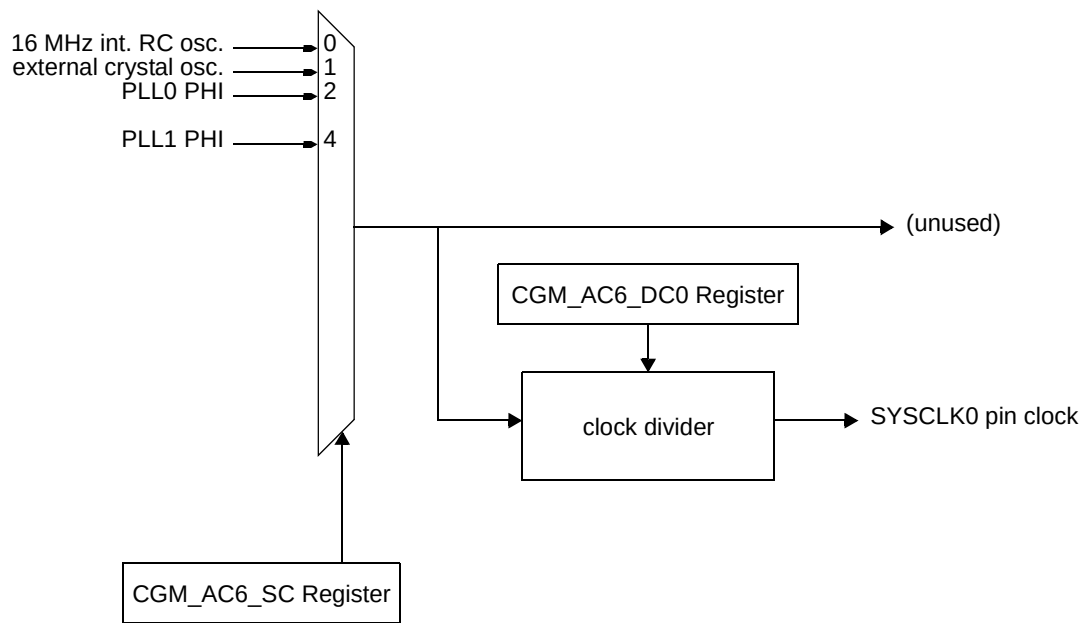


Figure 354. MC_CGM auxiliary clock 6 generation overview

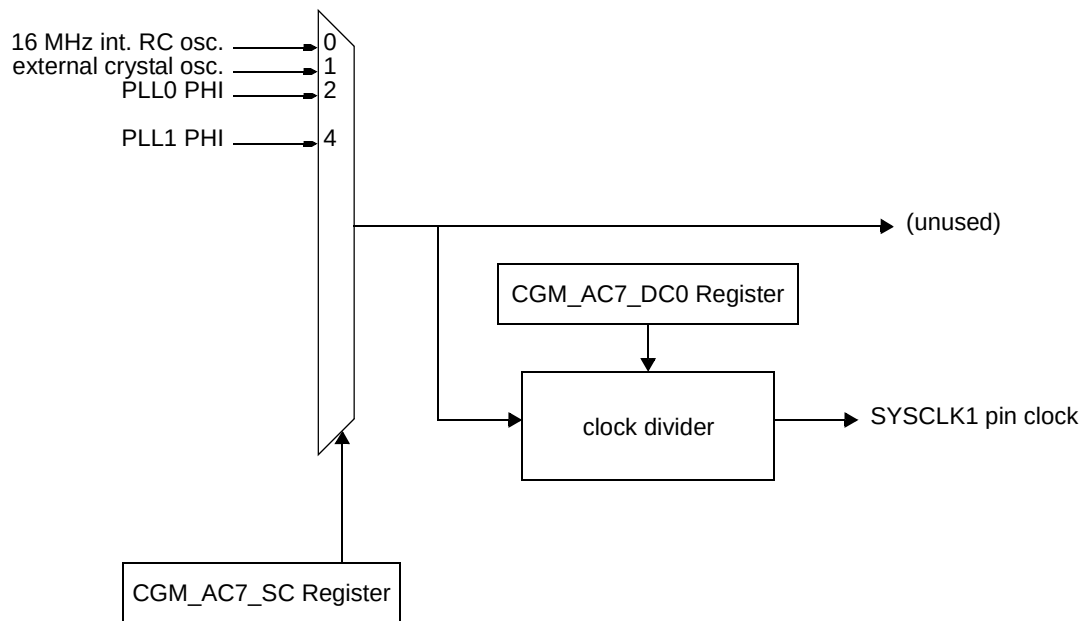


Figure 355. MC_CGM auxiliary clock 7 generation overview

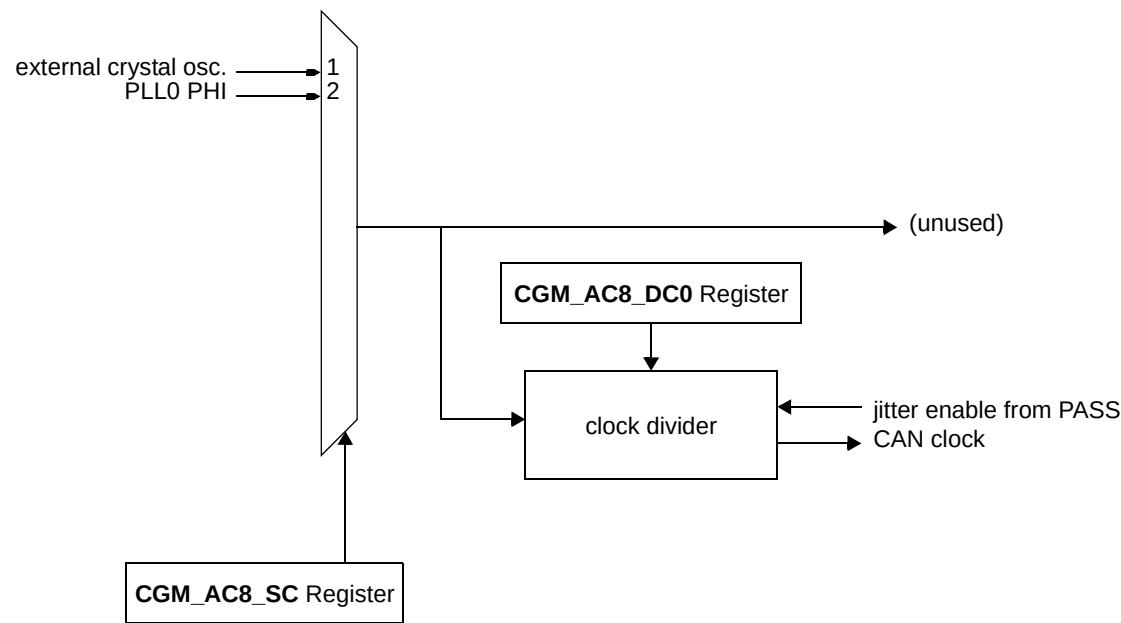


Figure 356. MC_CGM auxiliary clock 8 generation overview

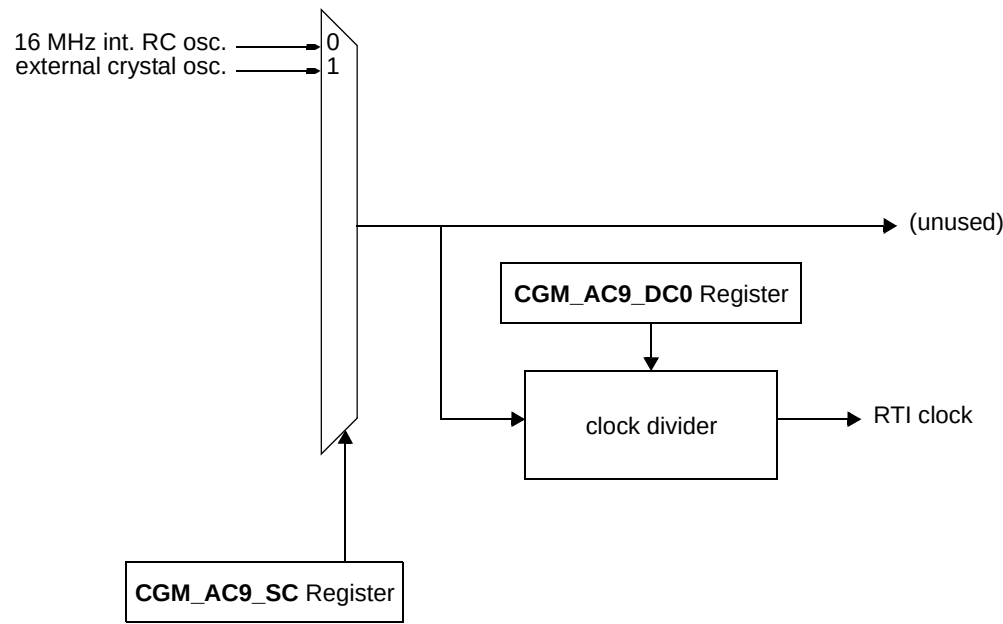


Figure 357. MC_CGM auxiliary clock 9 generation overview

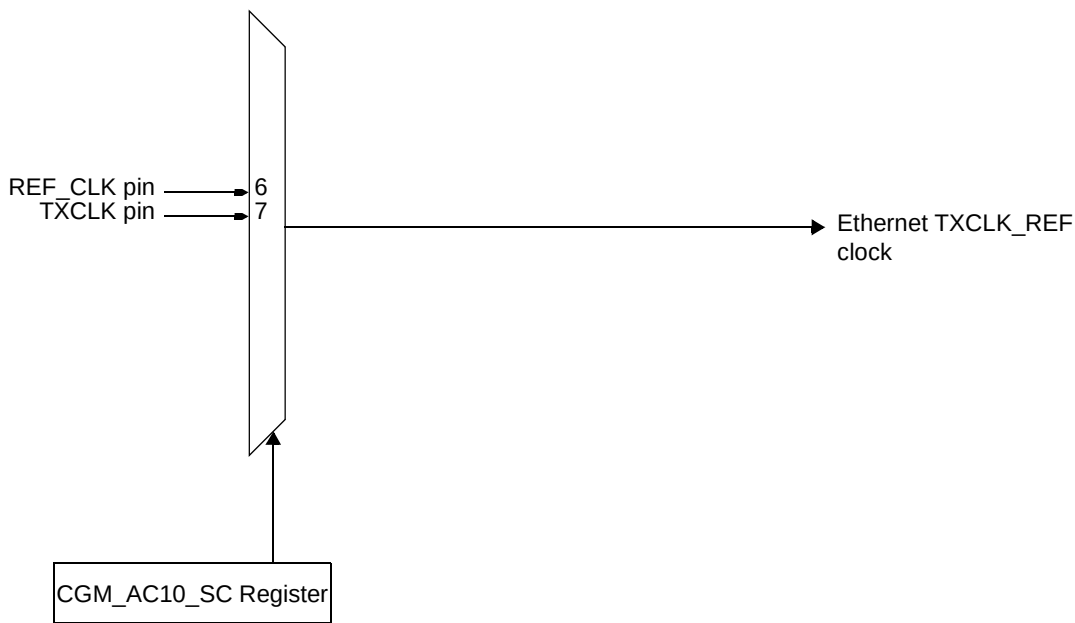


Figure 358. MC_CGM auxiliary clock 10 generation overview

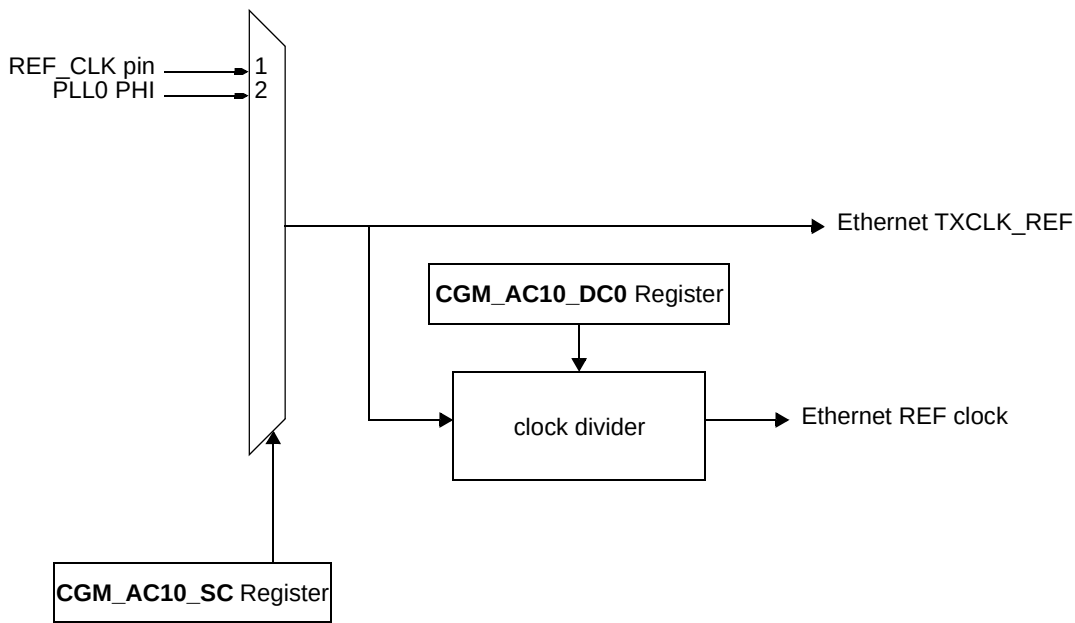


Figure 359. MC_CGM auxiliary clock 11 generation overview

30.4.2.1 Auxiliary clock dividers

The MC_CGM generates the following derived clocks:

- peripheral clock - controlled by the CGM_AC0_DC0 register
- sigma delta ADC clock - controlled by the CGM_AC0_DC1 register
- SAR ADC clock - controlled by the CGM_AC0_DC2 register
- DSPI clock 0 - controlled by the CGM_AC0_DC3 register
- LFAST clock - controlled by the CGM_AC1_DC0 register
- FlexRay clock - controlled by the CGM_AC2_DC0 register
- SENT clock - controlled by the CGM_AC2_DC1 register
- PSi5_f189 clock - controlled by the CGM_AC5_DC0 register
- PSi5_f125 clock - controlled by the CGM_AC5_DC1 register
- PSi5_1us clock - controlled by the CGM_AC5_DC2 register
- PS_S_TS clock - controlled by the CGM_AC5_DC2 and CGM_AC5_CDC0 registers
- PSi5_S_DDSR0 clock - controlled by the CGM_AC5_DC2, CGM_AC5_CDC1, and CGM_AC5_CDC10 registers
- PSi5_S_DDSR1 clock - controlled by the CGM_AC5_DC2, CGM_AC5_CDC1, and CGM_AC5_CDC11 registers
- PSi5_S_DDSR2 clock - controlled by the CGM_AC5_DC2, CGM_AC5_CDC1, and CGM_AC5_CDC12 registers
- PSi5_S_DDSR3 clock - controlled by the CGM_AC5_DC2, CGM_AC5_CDC1, and CGM_AC5_CDC13 registers
- PSi5_S_WD0 clock - controlled by the CGM_AC5_DC2, CGM_AC5_CDC2, and CGM_AC5_CDC20 registers
- PSi5_S_WD1 clock - controlled by the CGM_AC5_DC2, CGM_AC5_CDC2, and CGM_AC5_CDC21 registers
- PSi5_S_WD2 clock - controlled by the CGM_AC5_DC2, CGM_AC5_CDC2, and CGM_AC5_CDC22 registers
- PSi5_S_WD3 clock - controlled by the CGM_AC5_DC2, CGM_AC5_CDC2, and CGM_AC5_CDC23 registers
- SYSCLK0 pin clock - controlled by the CGM_AC6_DC0 register
- SYSCLK1 pin clock - controlled by the CGM_AC7_DC0 register
- CAN clock - controlled by the CGM_AC8_DC0 register
- RTI clock - controlled by the CGM_AC9_DC0 register
- ethernet REF clock - controlled by the CGM_AC10_DC0 register

30.4.3 Dividers functional description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- [Section 30.3.1.23, System Clock Divider 0 Configuration Register \(CGM_SC_DC0\)](#)
- [Section 30.3.1.24, System Clock Divider 1 Configuration Register \(CGM_SC_DC1\)](#)
- [Section 30.3.1.25, System Clock Divider 2 Configuration Register \(CGM_SC_DC2\)](#)
- [Section 30.3.1.28, Auxiliary Clock 0 Divider 0 Configuration Register \(CGM_AC0_DC0\)](#)
- [Section 30.3.1.29, Auxiliary Clock 0 Divider 1 Configuration Register \(CGM_AC0_DC1\)](#)
- [Section 30.3.1.30, Auxiliary Clock 0 Divider 2 Configuration Register \(CGM_AC0_DC2\)](#)
- [Section 30.3.1.31, Auxiliary Clock 0 Divider 3 Configuration Register \(CGM_AC0_DC3\)](#)
- [Section 30.3.1.33, Auxiliary Clock 1 Select Control Register \(CGM_AC1_SC\)](#)
- [Section 30.3.1.35, Auxiliary Clock 1 Divider 0 Configuration Register \(CGM_AC1_DC0\)](#)
- [Section 30.3.1.36, Auxiliary Clock 2 Divider 0 Configuration Register \(CGM_AC2_DC0\)](#)
- [Section 30.3.1.37, Auxiliary Clock 2 Divider 1 Configuration Register \(CGM_AC2_DC1\)](#)
- [Section 30.3.1.42, Auxiliary Clock 5 Divider 0 Configuration Register \(CGM_AC5_DC0\)](#)
- [Section 30.3.1.43, Auxiliary Clock 5 Divider 1 Configuration Register \(CGM_AC5_DC1\)](#)
- [Section 30.3.1.44, Auxiliary Clock 5 Divider 2 Configuration Register \(CGM_AC5_DC2\)](#)
- [Section 30.3.1.1, Auxiliary Clock 5 Cascaded Divider 0 Configuration Register \(CGM_AC5_CDC0\)](#)
- [Section 30.3.1.2, Auxiliary Clock 5 Cascaded Divider 1 Configuration Register \(CGM_AC5_CDC1\)](#)
- [Section 30.3.1.3, Auxiliary Clock 5 Cascaded Divider 2 Configuration Register \(CGM_AC5_CDC2\)](#)
- [Section 30.3.1.4, Auxiliary Clock 5 Cascaded Divider 10 Configuration Register \(CGM_AC5_CDC10\)](#)
- [Section 30.3.1.5, Auxiliary Clock 5 Cascaded Divider 11 Configuration Register \(CGM_AC5_CDC11\)](#)
- [Section 30.3.1.6, Auxiliary Clock 5 Cascaded Divider 12 Configuration Register \(CGM_AC5_CDC12\)](#)
- [Section 30.3.1.7, Auxiliary Clock 5 Cascaded Divider 13 Configuration Register \(CGM_AC5_CDC13\)](#)
- [Section 30.3.1.8, Auxiliary Clock 5 Cascaded Divider 20 Configuration Register \(CGM_AC5_CDC20\)](#)
- [Section 30.3.1.9, Auxiliary Clock 5 Cascaded Divider 21 Configuration Register \(CGM_AC5_CDC21\)](#)

- [Section 30.3.1.10, Auxiliary Clock 5 Cascaded Divider 22 Configuration Register \(CGM_AC5_CDC22\)](#)
- [Section 30.3.1.11, Auxiliary Clock 5 Cascaded Divider 23 Configuration Register \(CGM_AC5_CDC23\)](#)
- [Section 30.3.1.47, Auxiliary Clock 6 Divider 0 Configuration Register \(CGM_AC6_DC0\)](#)
- [Section 30.3.1.50, Auxiliary Clock 7 Divider 0 Configuration Register \(CGM_AC7_DC0\)](#)
- [Section 30.3.1.53, Auxiliary Clock 8 Divider 0 Configuration Register \(CGM_AC8_DC0\)](#)
- [Section 30.3.1.56, Auxiliary Clock 9 Divider 0 Configuration Register \(CGM_AC9_DC0\)](#)

The reset value of all counters is '1'. If a divider has its DE bit in the respective configuration register set to '0' (the divider is disabled), any value in its DIV field is ignored.

30.4.3.1 Divider jitter injection

The clock dividers for the FlexRay clock and the CAN clock inject a jitter so that off-chip communication via the associated peripherals is not possible. For details on enabling this feature, see the "Production Disable" and "CAN Jitter Enable" sections of the PASS chapter.

In addition, PLL0 PHI is always selected to source the FlexRay clock when jitter injection is enabled (see [Figure 350](#)).

30.4.3.2 Fractional divider details

The clock divider used to generate the PSI5_f189 clock and the DSPI clock 0 divides the selected clock source frequency by a fractional value given by the DIV and DIV_FMT fields of the CGM_AC5_DC0 register.

Equation 21
$$F_{\text{divided clock}} = F_{\text{source clock}} \div ((\text{DIV} + 1) \div 10^{\text{DIV_FMT}})$$

The divided clock frequency is actually an average frequency, since the divider uses only the available rising and falling edges of the source clock. Therefore, the actual frequency switches between two frequencies which are the two frequencies closest to the desired frequency (above and below) which can be extracted by a division factor $n/2$, where n is an integer greater than 1. The switching between the two frequencies is such that the resultant clock's rising edge never deviates from the ideal clock's rising edge by more than 25% of the source clock period and that the resultant average frequency over $\text{DIV} + 1$ source clock periods ($= 10^{\text{DIV_FMT}}$ divided clock periods) is the desired frequency.

30.4.3.2.1 Example: Fractional clock division

Desired division factor = 2.3

DIV = 22

DIV_FMT = 1

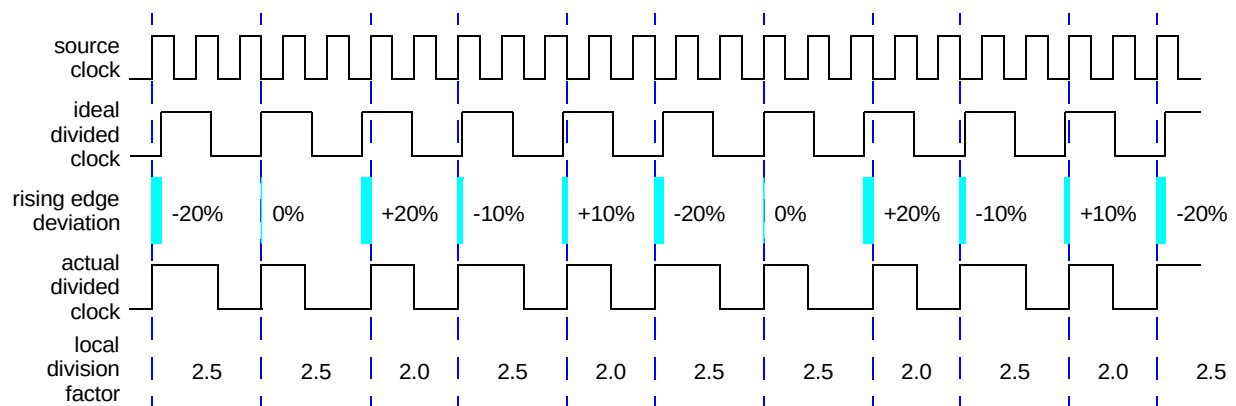


Figure 360. MC_CGM fractional division example waveform (divide by 2.3)

$$\text{Equation 22} \quad F_{\text{divided clock}}|_{\text{max}} = F_{\text{source clock}} \div 2.0$$

$$\text{Equation 23} \quad F_{\text{divided clock}}|_{\text{min}} = F_{\text{source clock}} \div 2.5$$

$$\text{Equation 24} \quad F_{\text{divided clock}}|_{\text{ave}} = F_{\text{source clock}} \div \frac{((2.5 \times 6) + (2.0 \times 4))}{10}$$

Therefore,

$$\text{Equation 25} \quad F_{\text{divided clock}}|_{\text{ave}} = F_{\text{source clock}} \div 2.3$$

As can be seen in [Figure 360](#), the rising edge of the actual divided clock is 'snapped' to the rising or falling edge of the source clock closest to the rising edge of the ideal divided clock. This results in a theoretical maximum deviation, or jitter, of $\pm 25\%$ of the source clock period. In this particular case, the maximum deviation is $\pm 20\%$ of the source clock period.

30.4.3.3 Divider Update Triggering

There are two methods for updating the dividers associated with the system clock and each auxiliary clock: immediate and register triggered. Which method is used for which set of dividers is controlled by the CGM_DIV_UPD_TYPE register.

Regardless of which method is selected for a given group of dividers, once an update of one of the dividers of that group has been triggered, the associated bit in the CGM_UPD_STAT register is set. This bit is then cleared when all the dividers of that group have completed the update process and are running with the current configuration.

30.4.3.3.1 Immediate Divider Update

If a divider group has been configured to be updated immediately (i.e., the corresponding CGM_DIV_UPD_TYPE register bit = '0'), the configuration of a divider in that group will be applied immediately on the write to that divider's configuration register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

30.4.3.3.2 Software-triggered divider update

If a divider group has been configured to be updated via a software trigger (i.e., the corresponding CGM_DIV_UPD_TYPE register bit = '1'), the configuration of a divider in that group will not be applied immediately on the write to that divider's configuration register. Instead, the configuration in the corresponding configuration register will be applied only when software writes any value to the CGM_DIV_UPD_TRIG register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

In the case of the system clock dividers, if the SYS_DIV_RATIO_CHNG bit in the CGM_SC_DIV_RC register = '1', the crossbar switch is requested to halt its ongoing transactions, and only after the crossbar switch has acknowledged the request does the update of the system clock dividers take place. Once these updates have completed, the halt request to the crossbar is deasserted.

Note: If halting the crossbar can cause a bus master to starve and thus lead to lost data or other problems, it is recommended to vary the system clock divider configurations without changing the ratios and to set CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] to '0'.

Note: In order to ensure a clean clock divider configuration update, software must check that all ongoing updates have already completed (i.e., all relevant bits in the CGM_DIV_UPD_STAT register are '0') before triggering a new update via the CGM_DIV_UPD_TRIG register.

31 OSC Digital Interface (XOSC)

31.1 Introduction

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provide the register interface for the programmable features. Selection of the XOSC as a source clock is controlled the Clock Generation module (MC_CGM). The XOSC is enabled and disabled by configuring the Mode Entry module (MC_ME).

The programmable features of the XOSC digital interface are as follows:

- Oscillator power-down control and status
- Oscillator startup delay
- Oscillator clock available interrupt
- Oscillator bypass mode control

31.2 Functional description

The XOSC digital interface provides control of the on-chip oscillator and the register interface for the mode and startup delay control. The oscillator provides an output clock that is used as an input to PLL0 and can be selected as the reference clock for many of the peripherals on the device.

[Table 433](#) shows the possible configurations of the XOSC.

Table 433. XOSC configurations

OSC EN	CTL[OSCBYP]	XTAL	EXTAL	XOSC Output	XOSC MODE
0	0	No crystal, high-impedance	No crystal, high-impedance	0	Power down, IDDQ
x	1	x	external clock	XTALOUT	Bypass, OSC Disabled
1	0	crystal	crystal	XTALOUT	Normal, OSC Enabled
		ground	external clock	XTALOUT	Normal, OSC Enabled

31.2.1 Oscillator power-down control and status

Enabling and disabling of the oscillator is done in the Mode Entry (MC_ME) module (see “Mode Entry Module (MC_ME)” chapter). The ME_<mode>_MC[XOSCON] controls the power-down of oscillator based on the current device mode (see “MC_ME Memory Map” table for available *modes*). The status of the XOSC is provided in the MC_ME_GS[S_XOSC] field.

31.2.2 Oscillator startup delay

A bit in the UTEST flash memory determines whether the oscillator is enabled at power-up of the device. If enabled, the oscillator is started after Phase3 of the device reset sequence. In order to allow the oscillator to reach full amplitude before use, the internal counter (OSCCNT) is started when the device exits reset. The counter is driven by the oscillator output clock. When the counter reaches XOSC_CTL[EOCV] × 512, a signal is sent to the

MC_ME module, allowing a mode switch to the oscillator (when used as a system clock or an input to PLL0).

The default value for this field after reset depends on implementation and can be found in the XOSC Device Configuration. After reset, software can clear or change the EOCV field in order to shorten or lengthen the delay until the oscillator is ready.

Note: It is advised that if the EOCV value is modified it should be done as early as possible in user code. This will ensure that the new value will be used before OSCCNT reaches the count value.

31.2.3 Oscillator clock available interrupt

An interrupt can be generated when the count value is reached ($OSCCNT = XOSC_CTL[EOCV] \times 512$). The `XOSC_CTL[I_OSC]` will be set and an interrupt will be generated if the interrupt mask is set (`XOSC_CTL[M_OSC] = 1`).

31.2.4 Oscillator bypass mode

The oscillator circuit can be bypassed by setting `XOSC_CTL[OSCBYP]`. This bit can only be set by software, and is only cleared by a system reset. In bypass mode, the oscillator is disabled and the input clock on the EXTAL pins is level-shifted and driven to the logic on the device. The bypass configuration is independent of the power-down mode of the oscillator.

31.3 Memory map and register definition

[Table 434](#) shows the memory map of the XOSC registers.

Table 434. XOSC memory map

Offset	Register	Location
00h	XOSC_CTL—XOSC Control Register	on page 1010

31.3.1 Register descriptions

31.3.1.1 XOSC control register (XOSC_CTL)

Figure 361. XOSC_CTL register

Offset 00h								Access: Read/Write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OSCIBYP ^{(1),(3)}	0	0	0	0	0	0	0	EOCV ⁽³⁾							
W																
Reset	—	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC ⁽³⁾	0	0	— ⁽⁴⁾	— ⁽⁴⁾	— ⁽⁴⁾	— ⁽⁴⁾	— ⁽⁴⁾	I_OSC ⁽²⁾	0	0	0	0	0	0	0
W																
Reset	—	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Software can read as well as set the OSCIBYP bit.
2. Software can read as well as clear the I_OSC bit.
3. See the Clocking chapter for reset value.
4. These bits can be written with any value, but writes are ignored. A read returns last written value.

Table 435. XOSC_CTL register field descriptions

Field	Description
0 OSCIBYP	Crystal Oscillator bypass. This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL is used as root clock.
1:7	Reserved
8:15 EOCV	End of Count Value. These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV×512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask. This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
17:23	Reserved

Table 435. XOSC_CTL register field descriptions (continued)

Field	Description
24 I_OSC	Crystal oscillator clock interrupt. This bit is set by hardware when OSCCNT counter reaches the count value $EOCV \times 512$. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.
25:31	Reserved

32 IRCOSC digital interface

The Internal RC Oscillator digital interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC), as well as an internal temperature sensor and an internal voltage regulator used to compensate external variations in temperature and voltage. It also provides access to trimming information used for safety implementations.

32.1 Functional description

The IRCOSC provides a high frequency (f_{MRC}) clock. After deassertion of reset, the output clock is available after a short time (see Electrical Specifications for timing information) which is required for stabilization. IRCOSC status indication is found in the MC_ME_GS[S_IRC] field (see Mode Entry Module in the device reference manual).

During power on reset sequence, native trimming bits, fine trimming bits, and TSENS code bits are initialized from the TEST flash memory. These bits can be accessed for safety reasons through the IRCOSC native trimming register (IRCOSC_NT). Based on native trimming, the main RC oscillator output frequency can be incrementally trimmed by IRCOSC_CTL[USER_TRIM[4:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with dfvar_SW, as defined in the datasheet.

32.2 Memory map and register definition

[Table 436](#) shows the memory map of the IRCOSC registers.

Table 436. IRCOSC memory map

Offset (hex)	Register	Access	Reset Value (hex)	Location
00	CTL—IRCOSC Control Register	R/W	0000_0000	32.2.1.1/1012
04	NT—IRCOSC Native Trimming Register	R	0000_0yyy ⁽¹⁾	32.2.1.2/1014
08	TT—IRCOSC Temperature Trimming Register	R	yyyy_yyyy ⁽¹⁾	32.2.1.3/1015

1. See the “Clocking” chapter for specific reset values.

32.2.1 Register descriptions

32.2.1.1 Control register (CTL)

The CTL contains user programmable parameters for user frequency trim, output IRCOSC divider and dynamic element matching in the IRCOSC.

Figure 362. Control register (CTL)

Offset 00h												Access: Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	— ⁽²⁾	0	0	0	USER_TRIM ⁽¹⁾				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	— ⁽²⁾	— ⁽²⁾	— ⁽²⁾	— ⁽²⁾	— ⁽²⁾	— ⁽²⁾	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. IRCOSC_CTL is writable only in supervisor mode.
2. These bits can be written with any value, but writes are ignored. A read returns last written value.

Table 437. CTL register field descriptions

Field	Description
0:10	Reserved
11:15 USER_TRIM	User trimming bits with respect to nominal factory frequency The MSB of the USER_TRIM bits is used to determine whether the frequency will be increased or decreased. If MSB = 0 then a change in USER_TRIM[3:0] will decrease the frequency, and likewise, if MSB = 1 then a change in USER_TRIM[3:0] will increase the frequency. Table 438 shows how the IRCOSC_CTL[USER_TRIM] field is used to trim the IRCOSC frequency.
16:31	Reserved

Table 438. IRCOSC_CTL[USER_TRIM] frequency trimming calculation

USER_TRIM[4:0] value	Frequency
....	
10011	$f + (3 \times \delta f_{\text{var_SW}})$
10010	$f + (2 \times \delta f_{\text{var_SW}})$
10001	$f + \delta f_{\text{var_SW}}$
10000	f
00000	f
00001	$f - \delta f_{\text{var_SW}}$
00010	$f - (2 \times \delta f_{\text{var_SW}})$
00011	$f - (3 \times \delta f_{\text{var_SW}})$
....	

Note: See the datasheet for $\delta f_{\text{var_SW}}$ value and min/max frequency variations.

32.2.1.2 Native trimming register (NT)

The NT register contains trim values loaded from TEST flash memory for the IRCOSC. Its status is enabled for voltage regulator and temperature sensor. This register is read only.

Figure 363. Native Trimming Register (NT)

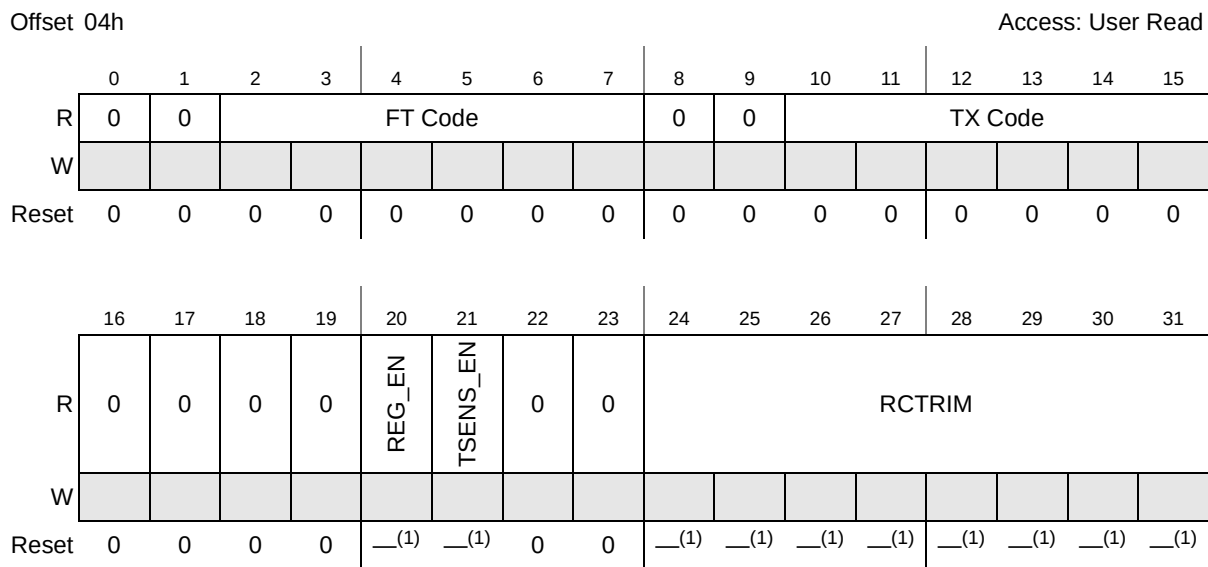


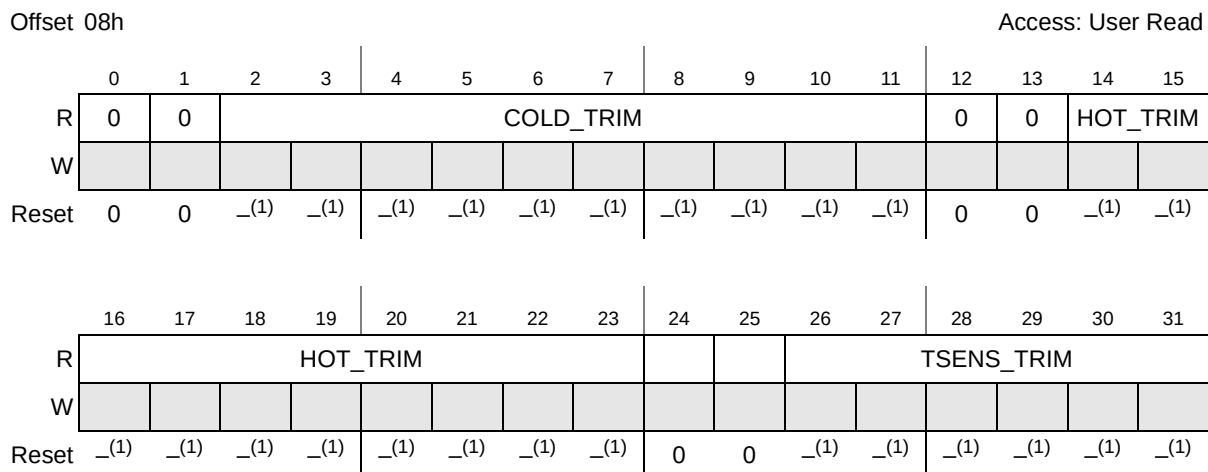
Table 439. NT register field descriptions

Field	Description
0:1	Reserved
2:7 FT Code	Output from INTERFACE inside RC to check the fine trimming code applied to RC_core
8:9	Reserved
10:15 TX Code	Output from TSENS inside RC to check the TSENS code
16:19	Reserved
20 REG_EN	Enables voltage regulator—trimming value for safety check 0 Voltage regulator is not enabled 1 Voltage regulator is enabled
21 TSENS_EN	Enables temperature sensor—trimming value for safety check 0 Temperature sensor is not enabled 1 Temperature sensor is enabled
22:23	Reserved
24:31 RCTRIM	IRCOSC trimming value for safety check

32.2.1.3 Temperature trimming register (TT)

The TT registers contains system trimming values for the IRCOSC that are loaded from TEST flash memory during reset. TT is a read only register.

Figure 364. Temperature Trimming Register (TT)



1. See Clocking chapter for specific reset values

Table 440. TT register field descriptions

Field	Description
0:1	Reserved
2:11 COLD_TRIM	Cold temperature trimming value for safety check
12:13	Reserved
14:23 HOT_TRIM	Hot temperature trimming value for safety check
24:25	Reserved
26:31 TSENS_TRIM	Ambient temperature reference code value for safety check

33 RAM controller (PRAM)

33.1 Introduction

This section provides an overview of the Platform RAM Controller. The RAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The RAM controller supports two 64-bit AHB interfaces and a 64-bit RAM array interface. The primary AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The backdoor AHB port provides a connection from the flash controller to facilitate calibration overlay access. [Figure 365](#) is a simplified block diagram depicting the position of the RAM controller within a typical platform architecture:

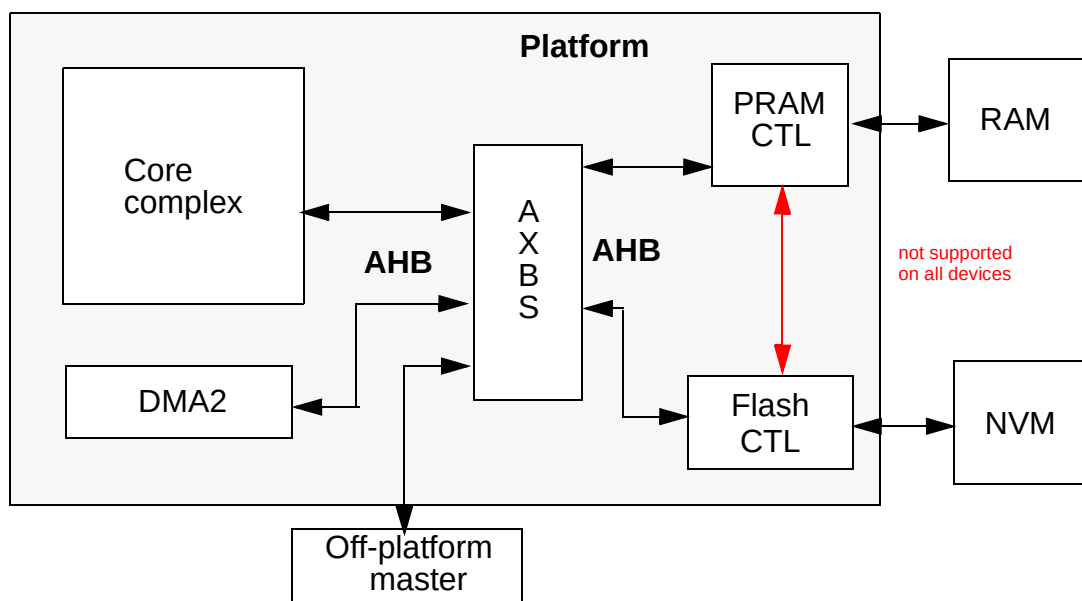


Figure 365. Simplified platform block diagram

The following list summarizes the key features of the RAM controller:

- System bus supports 64-bit data + 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data + 8-bit ECC interface
- Late-write support via 64-bit data + 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than a doubleword.

The address path of the RAM controller contains a mux that chooses among the addresses presented on the system bus for a read request, either the address from the temporary holding register or the address from the late-write buffer. The temporary holding register contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to

present simultaneously the address and associated write data, which is not valid until the AHB dataphase. The late-write buffer holds write requests which are delayed to facilitate single-cycle response on the system bus.

The read datapath contains a mux that chooses the source of the read data to be presented on the system bus on a read request. In the event that a read request matches the contents of the late-write buffer, a RAM access is not required and the late-write buffer contents are selected onto the read databus. Otherwise, the read data is a result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the RAM controller, including the late-write storage buffer, to support end-to-end ECC (e2eECC) coverage.

The write datapath contains the mux that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and instead be stored directly, precisely in the RAM.

33.2 SRAM controller memory map and register definition

The RAM controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses will generate an IPS error termination.

Table 441. RAM controller memory map

Offset	Register	Size	Access	Mode
PRAMC_BASE+00x00	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	32b	R/W	Supr ⁽¹⁾

1. Accessible only in supervisor mode. Accesses in user mode return a transfer error.

33.2.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)

The Platform RAM Configuration Register 1 (PRCR1) is used to specify operation of the RAM controller.

Note: *This register is accessible only in supervisor mode. Accesses in user mode returns a transfer error.*

Figure 366. Platform RAM Configuration Register 1 (PRCR1)

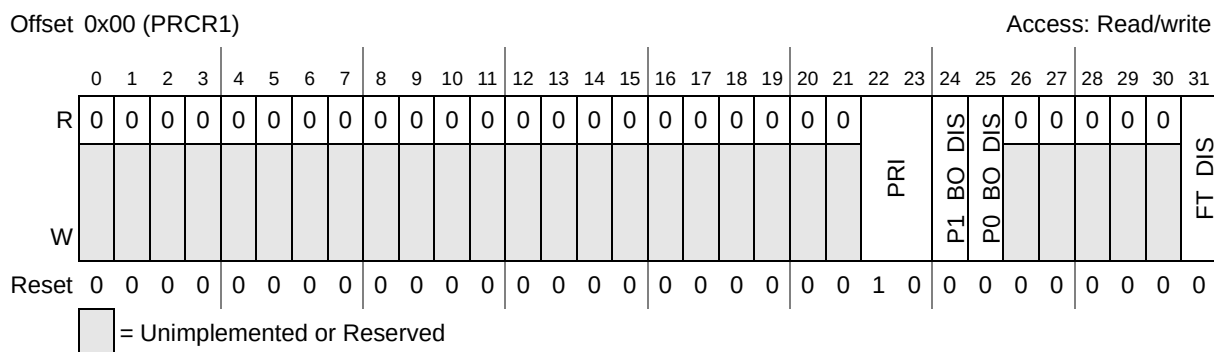


Table 442. Platform RAM Configuration Register 1 description

Field	Description
0 – 21	Reserved
22 – 23 PRI	AHB port arbitration mode 00 Round robin 01 Port p0 has priority over p1 10 Port p1 has priority over p0 11 Reserved
24 P1_BO_DIS	Port p1 read burst optimization disable 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response ⁽¹⁾ when PRCR[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when ⁽¹⁾ PRCR[FT_DIS]=1
25 P0_BO_DIS	Port p0 read burst optimization disable 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response ⁽¹⁾ when PRCR[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response ⁽¹⁾ when PRCR[FT_DIS]=1
26 – 30	Reserved
31 FT_DIS	Flow through disabled This field defines the AHB response of the RAM controller on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

1. The number of cycles taken for a RAM access can vary ± 1 clock cycle depending on the RAM speed relative to the PRAM controller clock frequency. If system RAM is running at the same frequency as the PRAM controller a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to PRAM controller clock frequency.

33.3 Functional description

33.3.1 Read / Write introduction

The RAM controller generates chip-select and write-enable, the array address, and write data as inputs to the RAM array. The RAM controller captures read data from the RAM array and drives it onto the AHB along with the associated datapath termination signals. To facilitate operating at higher frequencies, the RAM controller can optionally incur a single wait state on the AHB response.

33.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the AHB.

33.3.1.2 Optional read wait-state

The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCR1[FT_DIS] field inserts a register in the

read data path for use when operating the controller at high frequencies. The state of PRCR1[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCR1[FT_DIS]=0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if PRCR1[FT_DIS]=1, and a WRAP4 burst will have an access time of 3-2-2-2

Note: The number of cycles taken for a RAM access can vary ± 1 clock cycle depending on the RAM speed relative to the PRAM controller clock frequency. If system RAM is running at the same frequency as the PRAM controller a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to PRAM controller clock frequency.

33.3.2 Writes

33.3.2.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back-to-back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Rather, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

33.3.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with the potentially-corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. [Figure 367](#) provides details on the read-modify-write datapath. For details on the ECC coding scheme, refer to [Section 33.5, "Safety considerations."](#)

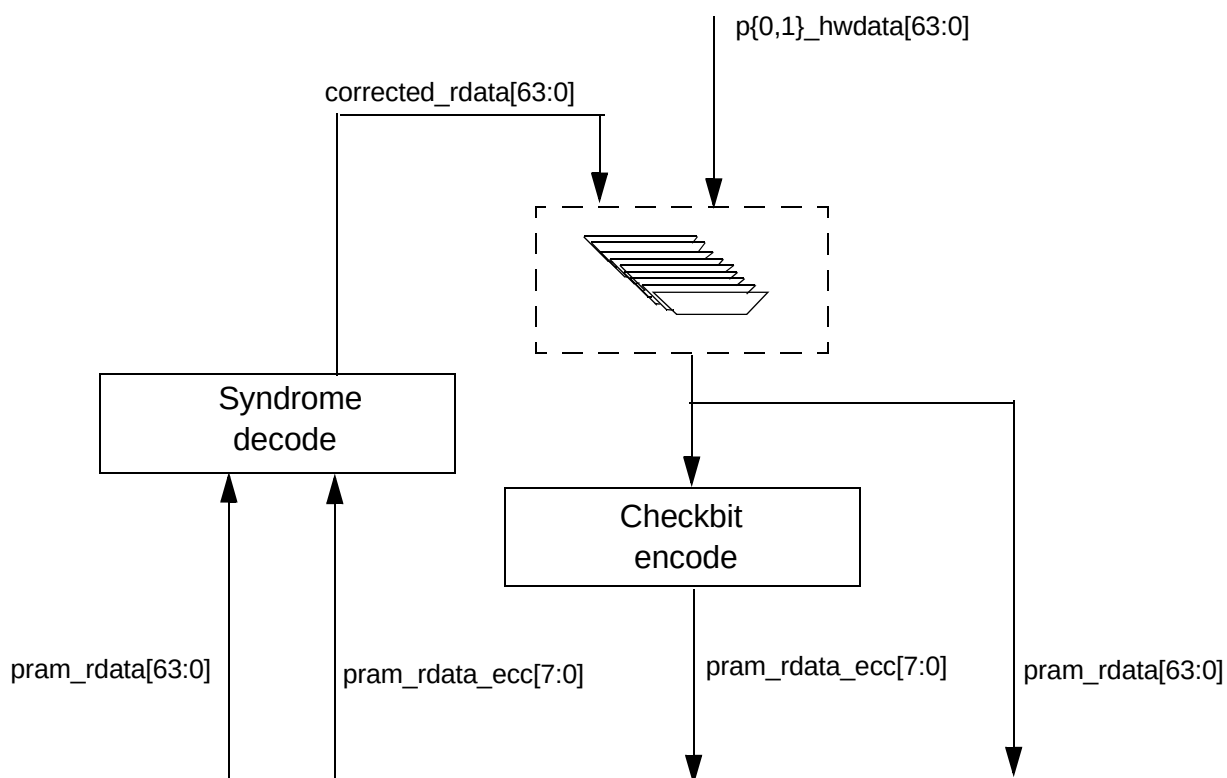


Figure 367. Read-modify-write datapath

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of size less than 64 bits require the insertion of one wait state before the data phase can be completed.

33.3.2.3 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

It should be noted that unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

33.4 Initialization/application information

It is essential that each memory address be written to a known value before it is read, to initialize the ECC. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Without writing an address to a known value first, a read from this address will most likely generate an uncorrectable ECC event.

A possibility for initializing PRAM memory space would be to use a stored 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

33.5 Safety considerations

33.5.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named for M.Y. Hsiao, an IBM researcher who published extensively in the early 1970s on SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns.
2. Every column is distinct.
3. Every column contains an odd number of ones, and hence is "odd weight".

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field).

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in [Table 443](#). A '*' in [Table 443](#) indicates the corresponding data or address bit is XOR'ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR'ed) together to the final value driven on the hwchckbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 443. RAM controller basic H-matrix definition

Checkbits [7:0]	Data Bit ⁽¹⁾																																
	Byte 7								Byte 6								Byte 5								Byte 4								
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
7	*				*					*			*		*	*		*	*	*	*					*	*				*	*	
6			*	*			*	*					*		*		*				*					*			*	*			*
5	*	*	*	*					*				*	*		*		*	*				*	*	*				*				
4	*		*			*		*					*		*		*			*			*	*			*			*			*
3									*	*	*	*	*		*				*		*					*	*	*	*				
2					*		*					*		*			*	*				*	*							*		*	
1		*		*		*		*	*	*	*	*					*		*		*		*							*		*	
0		*			*	*	*				*		*	*	*		*				*					*		*	*	*		*	

Table 443. RAM controller basic H-matrix definition (continued)

Checkbits [7:0]	Data Bit ⁽¹⁾																															
	Byte 3								Byte 2								Byte 1								Byte 0							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7					*		*			*		*					*	*			*	*			*				*	*		*
6	*	*		*	*				*			*					*	*		*	*					*	*				*	*
5			*	*			*	*	*			*	*	*					*		*	*	*	*								
4	*		*	*				*		*	*				*	*	*			*				*	*			*		*		
3	*	*	*			*					*	*		*	*			*					*		*	*	*			*		
2		*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
1		*						*		*		*	*	*			*				*				*		*		*	*	*	
0		*		*		*	*	*				*				*			*				*		*			*	*	*	*	*
Checkbits [7:0]	Address Bit ⁽¹⁾																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3			
7		*		*		*	*	*	*	*		*	*	*	*	*	*	*		*		*		*	*	*	*	*				
6		*		*	*	*	*	*		*	*		*	*	*	*	*	*		*	*	*	*	*	*	*	*	*				
5		*	*	*			*	*			*	*	*	*	*	*	*	*		*	*	*	*	*	*	*	*	*	*			
4	*	*	*		*	*	*	*		*	*	*	*	*	*				*	*	*	*	*	*	*	*	*	*	*			
3	*		*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*			
2	*	*			*		*	*	*	*	*			*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*			
1	*		*	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*		
0	*		*	*	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*		

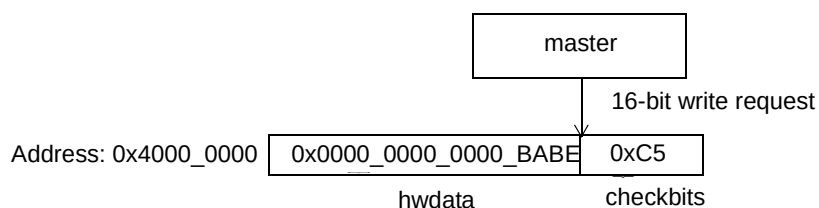
1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

33.5.1.1 E2EECC considerations on less-than-64-bit write transactions

Recall writes of size less than 64-bit incur a read-modify-write action as a consequence of the ECC coding scheme 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The RAM controller uses the following checkbit manipulation technique to ensure any faults in the PRAM control logic associated with performing the read-modify-write action will ultimately be detectable by e2eECC:

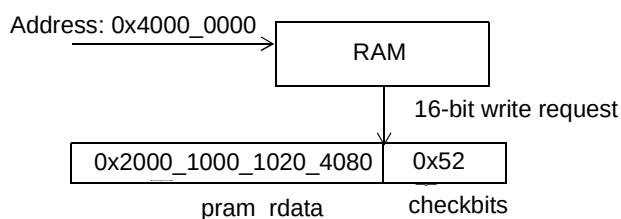
- On an 8-, 16-, or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non-pertinent byte lanes zeroed out, and the associated checkbit is based on this “zero-padded” write data.

Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

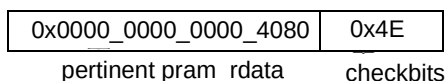


- The RAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

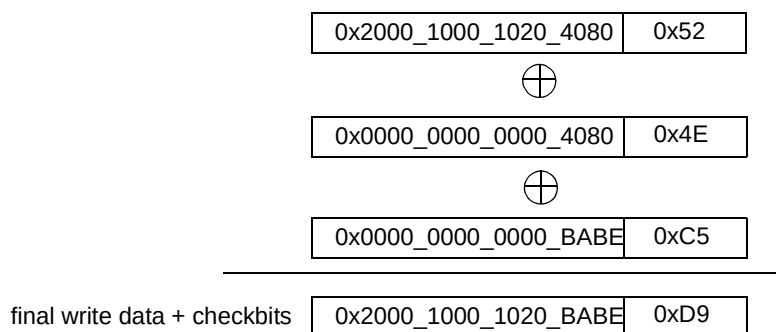
If a non-correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.



- The RAM controller isolates the pertinent byte lanes of the read data by zero-padding the non-pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.



- Merge the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and RAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The RAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

33.5.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by e2eECC check performed by the requesting master. Functional safety coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the RAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions

- Direct, system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as in indication of received inputs when a RAM access is initiated.

Table 444. RAM safety feature outputs

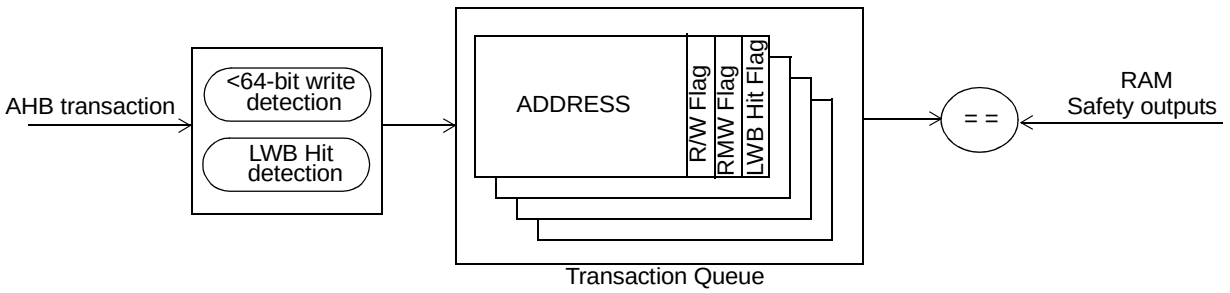
Signal Name	Description
pram_addr_co[17:0]	Safety feature compare outputs for address bus
pram_cs_b_co	Safety feature compare output for chip-select control input
pram_we_co	Safety feature compare output for write-enable control input

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, the safety feature compare outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

As a countermeasure against false misses to the late-write buffer on reads, the transaction monitor must also verify the expected source of data returned on a read as supplied either from the RAM or from the late-write buffer.

Figure 368. RAM controller transaction monitor block diagram



34 Flash memory controller

34.1 Introduction

The flash memory controller has these functions:

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash memory array.
- It serves as the interface to the on-chip overlay RAM.
- It serves as the interface to the off-chip buddy device.^(h)
- It acts as the interface between the ASC Modeling Unit (AMU) and the c55fm flash memory array.

The flash memory controller supports two 64-bit AHB buses and a 256-bit read data interface from each flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the flash array page size and providing 512 bytes of high-speed local storage. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage (e2eECC).

34.2 Features

The following list summarizes the key features of the flash memory controller:

- Two 64-bit AHB interface ports (p0, p1) allowing simultaneous access to dedicated prefetch mini-cache per slave port.
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle “buffer hit” read response.
- Configurable access control based on read/write and AHB master ID attributes.
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets.
- Optional address pipelining capability to maximize throughput.
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary.
- Configurable overlay remapping of logical flash accesses to on-chip calibration RAM, extended off-chip calibration RAM, on-chip system RAM
- Nexus trace stream interface supports message dumping to the on-chip overlay RAM
- Dedicated flash access port for servicing the AMU

h. The term “buddy device” refers to an optional companion chip with additional memory and debug features that is paired with a production chip during system development. It is not used in production devices.

34.3 Block diagrams

Figure 369 provides a block diagram showing the flash memory controller and the attached flash array.

Note: This module is also known as the Platform Flash controller, that is, PFlash or PFLASH_C55FM.

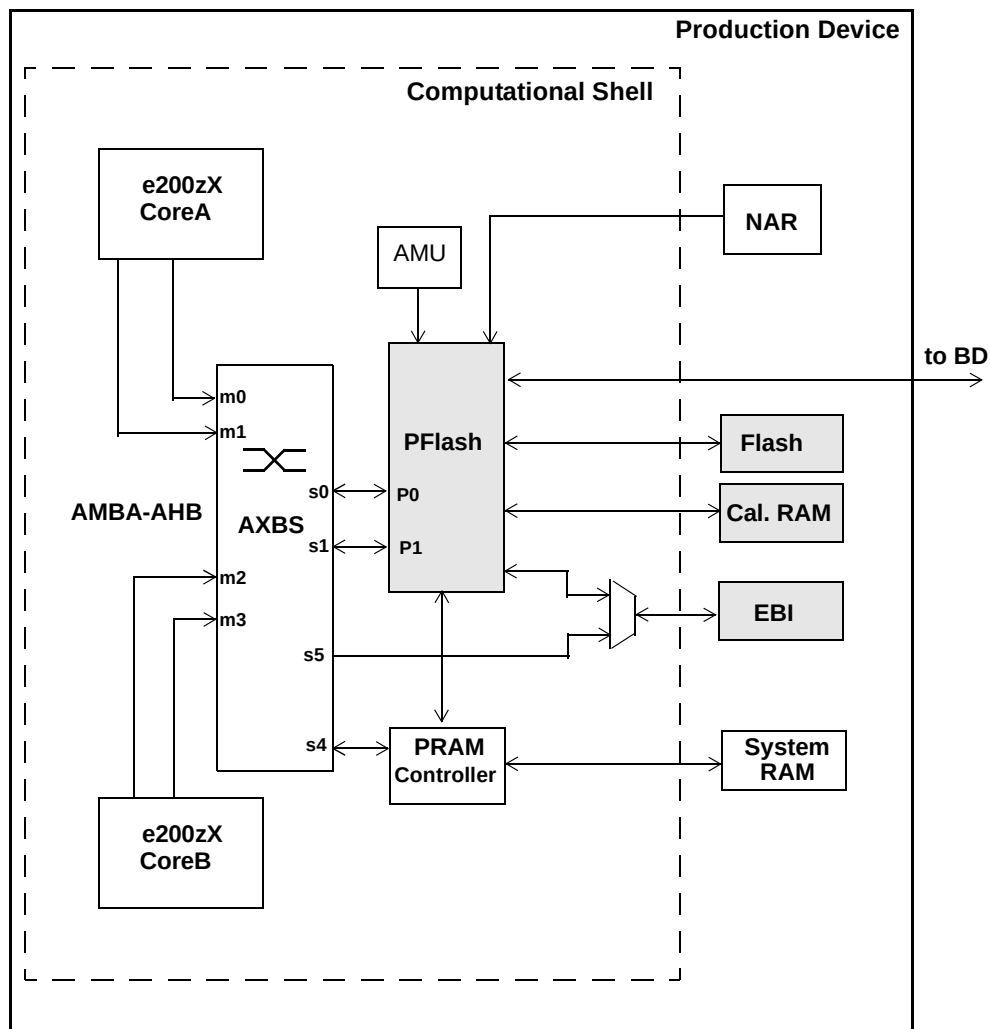


Figure 369. Platform-centric simple block diagram with flash memory controller

34.4 Flash memory controller memory map

34.4.1 Overview

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash memory access configuration and overlay remapping configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the flash memory controller module is in the midst of an operation will result in non-deterministic behavior. Software must be architected to avoid this scenario.

There is no idle indicator for the flash controller. The recommended flow for multi-core devices is to start only one core and execute initialization code to completion before starting the remaining cores. If the user needs to reconfigure the flash, code execution must be temporarily moved to system RAM.

Table 445. Flash memory controller memory map

Address Offset	Register	Access	Reset Value	Section/Page
00x00	Platform Flash Configuration Register 1 (PFCR1)	R/W	0x0000_0600	34.4.1.1/1030
0x004	Platform Flash Configuration Register 2 (PFCR2)	R/W	0x0000_0000	34.4.1.2/1033
0x008	Platform Flash Configuration Register 3 (PFCR3)	R/W	0x0000_0000	34.4.1.3/1037
0x00c	Platform Flash Access Protection Register (PFAPR)	R/W	0x1111_1111	34.4.1.4/1040
0x010	Platform Flash Remap Configuration Register (PFCRCR)	R/W	0x0000_0000	34.4.1.5/1043
0x014	Platform Flash Remap Descriptor Enable Register (PFCRDE)	R/W	0x0000_0000	34.4.1.6/1044
0x100	Platform Flash Calibration Region Descriptor 0 (PFCRD0)	R/W	0x0000_0000	34.4.1.7/1051
0x110	Platform Flash Calibration Region Descriptor 1 (PFCRD1)	R/W	0x0000_0000	34.4.1.7/1051
0x120	Platform Flash Calibration Region Descriptor 2 (PFCRD2)	R/W	0x0000_0000	34.4.1.7/1051
0x130	Platform Flash Calibration Region Descriptor 3 (PFCRD3)	R/W	0x0000_0000	34.4.1.7/1051
0x140	Platform Flash Calibration Region Descriptor 4 (PFCRD4)	R/W	0x0000_0000	34.4.1.7/1051
0x150	Platform Flash Calibration Region Descriptor 5 (PFCRD5)	R/W	0x0000_0000	34.4.1.7/1051
0x160	Platform Flash Calibration Region Descriptor 6 (PFCRD6)	R/W	0x0000_0000	34.4.1.7/1051
0x170	Platform Flash Calibration Region Descriptor 7 (PFCRD7)	R/W	0x0000_0000	34.4.1.7/1051
0x180	Platform Flash Calibration Region Descriptor 8 (PFCRD8)	R/W	0x0000_0000	34.4.1.7/1051
0x190	Platform Flash Calibration Region Descriptor 9 (PFCRD9)	R/W	0x0000_0000	34.4.1.7/1051
0x1A0	Platform Flash Calibration Region Descriptor 10 (PFCRD10)	R/W	0x0000_0000	34.4.1.7/1051
0x1B0	Platform Flash Calibration Region Descriptor 11 (PFCRD11)	R/W	0x0000_0000	34.4.1.7/1051

Table 445. Flash memory controller memory map (continued)

Address Offset	Register	Access	Reset Value	Section/Page
0x1C0	Platform Flash Calibration Region Descriptor 12 (PFCRD12)	R/W	0x0000_0000	34.4.1.7/1051
0x1D0	Platform Flash Calibration Region Descriptor 13 (PFCRD13)	R/W	0x0000_0000	34.4.1.7/1051
0x1E0	Platform Flash Calibration Region Descriptor 14 (PFCRD14)	R/W	0x0000_0000	34.4.1.7/1051
0x1F0	Platform Flash Calibration Region Descriptor 15 (PFCRD15)	R/W	0x0000_0000	34.4.1.7/1051
0x200	Platform Flash Calibration Region Descriptor 16 (PFCRD16)	R/W	0x0000_0000	34.4.1.7/1051
0x210	Platform Flash Calibration Region Descriptor 17 (PFCRD17)	R/W	0x0000_0000	34.4.1.7/1051
0x220	Platform Flash Calibration Region Descriptor 18 (PFCRD18)	R/W	0x0000_0000	34.4.1.7/1051
0x230	Platform Flash Calibration Region Descriptor 19 (PFCRD19)	R/W	0x0000_0000	34.4.1.7/1051
0x240	Platform Flash Calibration Region Descriptor 20 (PFCRD20)	R/W	0x0000_0000	34.4.1.7/1051
0x250	Platform Flash Calibration Region Descriptor 21 (PFCRD21)	R/W	0x0000_0000	34.4.1.7/1051
0x260	Platform Flash Calibration Region Descriptor 22 (PFCRD22)	R/W	0x0000_0000	34.4.1.7/1051
0x270	Platform Flash Calibration Region Descriptor 23 (PFCRD23)	R/W	0x0000_0000	34.4.1.7/1051
0x280	Platform Flash Calibration Region Descriptor 24 (PFCRD24)	R/W	0x0000_0000	34.4.1.7/1051
0x290	Platform Flash Calibration Region Descriptor 25 (PFCRD25)	R/W	0x0000_0000	34.4.1.7/1051
0x2A0	Platform Flash Calibration Region Descriptor 26 (PFCRD26)	R/W	0x0000_0000	34.4.1.7/1051
0x2B0	Platform Flash Calibration Region Descriptor 27 (PFCRD27)	R/W	0x0000_0000	34.4.1.7/1051
0x2C0	Platform Flash Calibration Region Descriptor 28 (PFCRD28)	R/W	0x0000_0000	34.4.1.7/1051
0x2D0	Platform Flash Calibration Region Descriptor 29 (PFCRD29)	R/W	0x0000_0000	34.4.1.7/1051
0x2E0	Platform Flash Calibration Region Descriptor 30 (PFCRD30)	R/W	0x0000_0000	34.4.1.7/1051
0x2F0	Platform Flash Calibration Region Descriptor 31 (PFCRD31)	R/W	0x0000_0000	34.4.1.7/1051

34.4.1.1 Platform Flash Configuration Register 1 (PFCR1)

The PFlash Configuration Register 1 (PFCR1) is used to specify operation Port p0 of the flash memory controller.

Figure 370. PFCR1 register

Offset 0x000																Access: Supervisor	
																Read/Write	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE	
W	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	APC				RWSC				0	P0_DPFE	0	P0_IPFE	0	P0_PFLIM		P0_BFEN	
W	APC				RWSC					P0_DPFE		P0_IPFE		P0_PFLIM		P0_BFEN	
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1

Note: See the Device Configuration chapter for details about the actual masters available on the device.

Table 446. PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 446. PFCR1 field descriptions (continued)

Field	Description
4 P0_M11PFE	Port0 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P0_M4PFE	Port0 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 446. PFCR1 field descriptions (continued)

Field	Description
13 P0_M2PFE	Port0 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–18 APC	Address Pipeline Control. This field controls the number of cycles that a subsequent flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1. 000 Pipelined access to the flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 A pipelined access can be initiated 2 cycles before the previous data is valid 011 A pipelined access can be initiated 3 cycles before the previous data is valid 1-- Pipelined access to the flash is disabled and one wait state is inserted before a subsequent access can be initiated
19–23 RWSC	Read Wait State Control. This field controls the number of wait-states to be added to the best-case flash array access time for reads. The best-case flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the flash memory controller and the actual read access time of the flash memory controller. The required settings are documented in the SoC specification. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to 5'b00110 by hardware reset. This field applies to the configuration of Port0 and Port1. 00000 No additional wait-states are added 00001 One additional wait-state is added ... 11111 Thirty-one additional wait-states are added

Table 446. PFCR1 field descriptions (continued)

Field	Description
25 P0_DPFEN	Port0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
27 P0_IPFEN	Port0 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
29–30 P0_PFLIM	Port0 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 1x The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

34.4.1.2 Platform Flash Configuration Register 2 (PFCR2)

The PFlash Configuration Register 2 (PFCR2) specifies operation Port1 of the PFLASH_C55FM flash memory controller.

Figure 371. PFCR2 register

Offset 0x004

Access: Supervisor
Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P1_M15PFE	P1_M14PFE	P1_M13PFE	P1_M12PFE	P1_M11PFE	P1_M10PFE	P1_M9PFE	P1_M8PFE	P1_M7PFE	P1_M6PFE	P1_M5PFE	P1_M4PFE	P1_M3PFE	P1_M2PFE	P1_M1PFE	P1_M0PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	P1_DPFE	0	P1_IPFE	0	P1_PFLIM		P1_BFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Note: See the Device Configuration chapter for details about the actual masters available on the device.

Table 447. PFCR2 field descriptions

Name	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 447. PFCR2 field descriptions (continued)

Name	Description
4 P1_M11PFE	Port1 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P1_M10PFE	Port1 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P1_M5PFE	Port1 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P1_M4PFE	Port1 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 447. PFCR2 field descriptions (continued)

Name	Description
13 P1_M2PFE	Port1 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P1_M1PFE	Port1 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
25 P1_DPFEN	Port1 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
27 P1_IPFEN	Port1 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
29–30 P1_PFLIM	Port1 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 1x The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

34.4.1.3 Platform Flash Configuration Register 3 (PFCR3)

Figure 372. PFCR3 register

Offset 0x008												Access: Supervisor Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_WCFG		P1_WCFG		0	0	0	0	0	0	0	BDRM	0	0	0	BAF_DIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ARBM		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 448. PFCR3 field descriptions

Name	Description
0–1 P0_WCFG	Port0 Way Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers in an indexed set are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers in an indexed set are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved.

Table 448. PFCR3 field descriptions (continued)

Name	Description
2–3 P1_WCFG	Port1 Line Buffer Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with buffer 0 allocated for instruction fetches and buffer 1 for data accesses. 11 Reserved
11 BDRM	Buddy Device Read Mode - This field controls the PFLASH_C55FM bus response when the master presents a read request targeting the extended overlay RAM on the Buddy Device (BD) using the direct, non-remapped address. This field is ignored on production ECUs where the Buddy Device is not present. On production ECUs, this memory map region is provided for the “OLDA” (on-line data acquisition) function. Read requests on production ECUs targeting the non-existent extended overlay RAM using the direct, non-remapped address are terminated with error on the system bus. This field is ignored on write requests targeting the extended overlay RAM using the direct, non-remapped address. Write requests on production ECUs targeting the non-existent extended overlay RAM using the direct, non-remapped address and terminate cleanly on the system bus. Write requests on engineering ECUs targeting the extended overlay RAM using the direct, non-remapped address are ignored are executed. This field is initialized by hardware reset. 0 Read request targeting the extended overlay RAM on the Buddy Device (BD) using the direct, non-remapped address are aborted, and the request is terminated with error on the system bus. 1 Read request targeting the extended overlay RAM on the Buddy Device (BD) using the direct, non-remapped address are allowed. The PFLASH_C55FM returns the requested data from the extended overlay RAM, which resides on the BD. Any system bus errors returned by the BD are passed onto the master.

Table 448. PFCR3 field descriptions (continued)

Name	Description
15 BAF_DIS	BAF Disable. This field controls executable access to the BAF (Boot Assist Flash) region of the flash. Once this field is set, attempted instruction accesses targeting the BAF region are aborted and terminated with a system bus error. The affect of this field applies to system bus transfers through both, Port0 and Port1. Data-type accesses to the BAF region are not affected by this field. Once this field is set, it becomes a read-only field and can only be cleared by hardware reset. Once this field is set, any subsequent write attempts to modify this field are ignored with an error-free data transfer termination. This field is initialized by hardware reset. 0 Executable access to the BAF flash region is allowed. 1 Executable access to the BAF flash region is prohibited.
16–17 ARBM	Arbitration Mode. This 2-bit field controls the arbitration of concurrent flash access requests from the two AHB ports of the flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever both ports issue concurrent requests. This field is initialized by hardware reset. 00 Fixed priority arbitration with AHB p0>p1 01 Fixed priority arbitration with AHB p1>p0 1- Round-robin arbitration

34.4.1.4 Platform Flash Access Protection Register (PFAPR)

The PFlash Access Protection Register (PFAPR) is used to control read and write accesses to the flash array.

Figure 373. PFAPR register

Offset 0x00C												Access: Supervisor Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Note: See the *Device Configuration chapter* for details about the actual masters available on the device.

Table 449. PFlash Access Protection Register field descriptions

Name	Description
0–1 M0AP	Master 0 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2–3 M1AP	Master 1 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4–5 M2AP	Master 2 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 449. PFlash Access Protection Register field descriptions (continued)

Name	Description
6–7 M3AP	Master 3 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8–9 M4AP	Master 4 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10–11 M5AP	Master 5 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12–13 M6AP	Master 6 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
14–15 M7AP	Master 7 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16–17 M8AP	Master 8 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 449. PFlash Access Protection Register field descriptions (continued)

Name	Description
18–19 M9AP	Master 9 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20–21 M10AP	Master 10 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22–23 M11AP	Master 11 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
24–25 M12AP	Master 12 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26–27 M13AP	Master 13 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 449. PFlash Access Protection Register field descriptions (continued)

Name	Description
28–29 M14AP	Master 14 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30–31 M15AP	Master 15 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

34.4.1.5 PFlash Remap Control Register (PFCRCR)

The PFCRCR is used to globally enable/disable the calibration remap function.

Figure 374. PFCRCR register

Offset 0x010												Access: Supervisor				
Read/Write																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	SAFE_CAL	0	0	0	IRMEN	0	0	0	GRMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 450. PFCRCR field descriptions

Field	Description
23 SAFE_CAL	Safe Calibration. This bit facilitates the use of safety-critical calibration data. When this bit is enabled, the flash memory controller is configured to perform redundancy checking on the calibration remap function and reduce the total number of potential calibration regions from 32 to 16. This field is initialized by hardware reset. 0 Established calibration overlay regions are considered safety-critical. 1 Established calibration overlay regions are not considered safety-critical.
27 IRMEN	Instruction Remap. This bit enables calibration remapping evaluation on instruction fetches. When PFCRCR[GRMEN] is disabled, this bit is ignored. This field is initialized by hardware reset. 0 Calibration remap evaluation is performed on any incoming data fetch requests only. 1 Calibration remap evaluation is performed on all incoming flash access requests - data and instruction fetches.
31 GRMEN	Global Remap Enable. This bit globally enables or disables the calibration remap evaluation on system flash accesses. Calibration remap evaluation is immediately available the next cycle following an update to enable PFCRCR[GRMEN] This field is initialized by hardware reset. 0 Calibration remap evaluation is not performed on any incoming system flash access requests. 1 Calibration remap evaluation is performed on all incoming system flash access requests.

34.4.1.6 PFlash Remap Descriptor Enable Register (PFCRDE)

The PFlash Calibration Remap Descriptor Enable Register (PFCRDE) is used to enable or disable up to 32 calibration remap descriptors. Note there is also a global remap enable (PFCRCR[GRMEN]) that also must be asserted in conjunction with the individual CRDnEN flags to enable a given remap descriptor.

Figure 375. PFCRDE register

Offset 0x014

Access: Supervisor
Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRD0EN	CRD1EN	CRD2EN	CRD3EN	CRD4EN	CRD5EN	CRD6EN	CRD7EN	CRD8EN	CRD9EN	CRD10EN	CRD11EN	CRD12EN	CRD13EN	CRD14EN	CRD15EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRD16EN	CRD17EN	CRD18EN	CRD19EN	CRD20EN	CRD21EN	CRD22EN	CRD23EN	CRD24EN	CRD25EN	CRD26EN	CRD27EN	CRD28EN	CRD29EN	CRD30EN	CRD31EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 451. PFCRDE field descriptions

Field	Description
0 CRD0EN	Calibration Remap Descriptor 0 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
1 CRD1EN	Calibration Remap Descriptor 1 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
2 CRD2EN	Calibration Remap Descriptor 2 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other type of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid

Table 451. PFCRDE field descriptions (continued)

Field	Description
3 CRD3EN	Calibration Remap Descriptor 3 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
4 CRD4EN	Calibration Remap Descriptor 4 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
5 CRD5EN	Calibration Remap Descriptor 5 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
6 CRD6EN	Calibration Remap Descriptor 6 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
7 CRD7EN	Calibration Remap Descriptor 7 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid

Table 451. PFCRDE field descriptions (continued)

Field	Description
8 CRD8EN	Calibration Remap Descriptor 8 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
9 CRD9EN	Calibration Remap Descriptor 9 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
10 CRD10EN	Calibration Remap Descriptor 10 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
11 CRD11EN	Calibration Remap Descriptor 11 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
12 CRD12EN	Calibration Remap Descriptor 12 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid

Table 451. PFCRDE field descriptions (continued)

Field	Description
13 CRD13EN	Calibration Remap Descriptor 13 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
14 CRD14EN	Calibration Remap Descriptor 14 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
15 CRD15EN	Calibration Remap Descriptor 15 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
16 CRD16EN	Calibration Remap Descriptor 16 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
17 CRD17EN	Calibration Remap Descriptor 17 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid

Table 451. PFCRDE field descriptions (continued)

Field	Description
18 CRD18EN	Calibration Remap Descriptor 18 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
19 CRD19EN	Calibration Remap Descriptor 19 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
20 CRD20EN	Calibration Remap Descriptor 20 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
21 CRD21EN	Calibration Remap Descriptor 21 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
22 CRD22EN	Calibration Remap Descriptor 22 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid

Table 451. PFCRDE field descriptions (continued)

Field	Description
23 CRD23EN	Calibration Remap Descriptor 23 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
24 CRD24EN	Calibration Remap Descriptor 24 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
25 CRD25EN	Calibration Remap Descriptor 25 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
26 CRD26EN	Calibration Remap Descriptor 26 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
27 CRD27EN	Calibration Remap Descriptor 27 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid

Table 451. PFCRDE field descriptions (continued)

Field	Description
28 CRD28EN	Calibration Remap Descriptor 28 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
29 CRD29EN	Calibration Remap Descriptor 29 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
30 CRD30EN	Calibration Remap Descriptor 30 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid
31 CRD31EN	Calibration Remap Descriptor 31 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor <i>n</i> invalid 1 Calibration Remap Descriptor <i>n</i> valid

34.4.1.7 PFlash Calibration Region Descriptor *n* (PFCRDn)

Each 96-bit (12-byte) region descriptor specifies an overlay region where a flash access can be remapped during calibration and debug. The calibration remap descriptors are organized sequentially as 128-bit (16-byte) structures in the Platform Flash Controller's programming model. Each of the three 32-bit words that define a single calibration region are detailed in the subsequent sections; the fourth word is unused.

34.4.1.7.1 PFlash Calibration Region Descriptor *n*, Word0 (PFCRDn.Word0)

The first word of the flash memory controller overlay region descriptor defines the 0-modulo-size logical start (byte) address of the calibration remap region. It is software's responsibility

to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Successful writes to this word clear the calibration remap descriptor's valid bit.

Figure 376. PFlash Calibration Region Descriptor *n*, Word0 (PFCRD*n*.Word0)

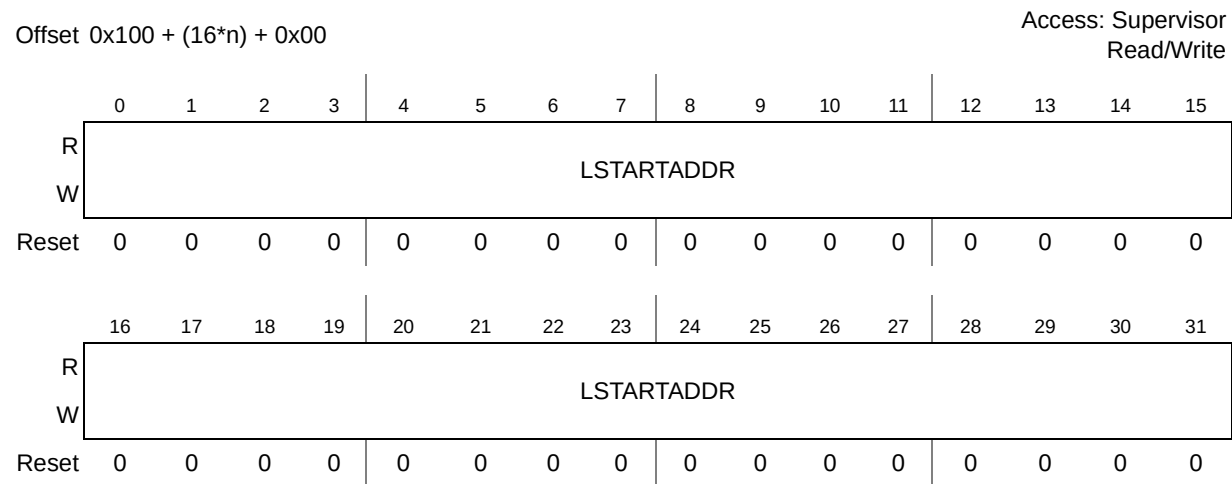


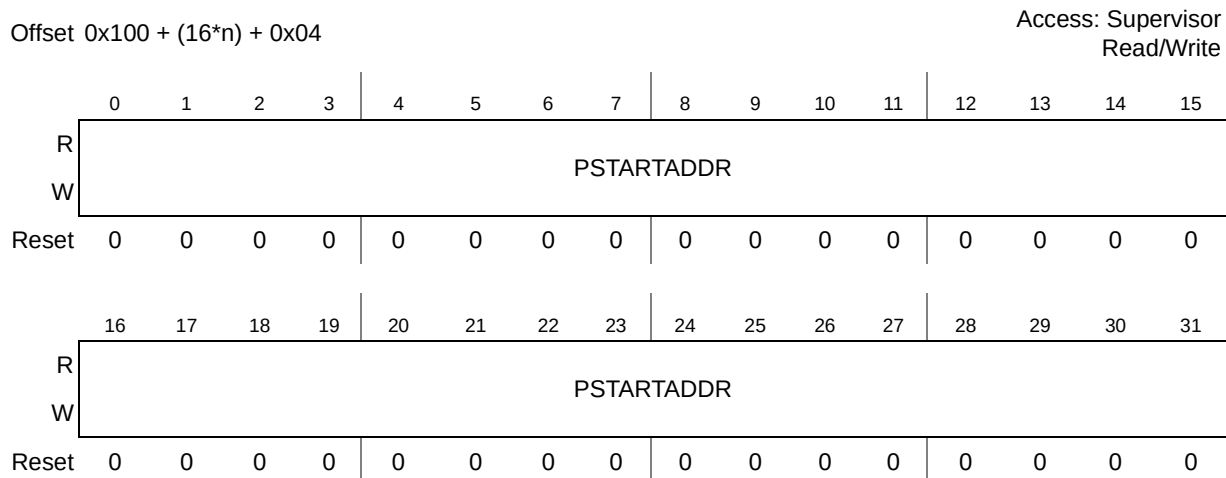
Table 452. PFlash Calibration Region Descriptor *n*, Word 0 description

Name	Description
0–27 LSTARTADDR	Logical Start Address. This field defines the most significant bits of the 0-modulo-size logical start address of the overlay remap region. It corresponds to the logical system address which maps to the flash memory space. Any write to PFCRD <i>n</i> .Word{0,1,2} clears the corresponding PFCRDE[CRD <i>n</i> EN] bit, leaving the calibration remap descriptor invalid.

34.4.1.7.2 PFlash Calibration Region Descriptor *n*, Word1 (PFCRD*n*.Word1)

The second word of the flash memory controller calibration region descriptor defines the 0-modulo-size byte physical start (byte) address of the calibration remap region. *The contents of this word define the targeted destination overlay memory.* It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Successful writes to this word clear the calibration remap descriptor's valid bit.

Figure 377. PFlash Calibration Region Descriptor n , Word1 (PFCRD n .Word1)Table 453. PFlash Overlay Region Descriptor n , Word 1 description

Name	Description
0–27 PSTARTADDR	Calibration Remap Descriptor n Physical Start Address - This field defines the most significant bits of the 0-modulo-size physical start byte address of the calibration remap descriptor. This address corresponds to the physical address which maps to the destination calibration overlay memory. Any write to PFCRD n .Word{0,1,2} clears the corresponding PFCRDE[CRD n EN] bit, leaving the calibration remap descriptor invalid.

34.4.1.7.3 PFlash Calibration Region Descriptor n , Word2 (PFCRD n .Word2)

The third word of the calibration region descriptor defines a per-master calibration remap enable and the remap region size. For cacheable spaces being remapped, the minimum region size is 32 bytes to match the flash page and cache line sizes. Writes to this word clear the calibration remap descriptor's valid bit.

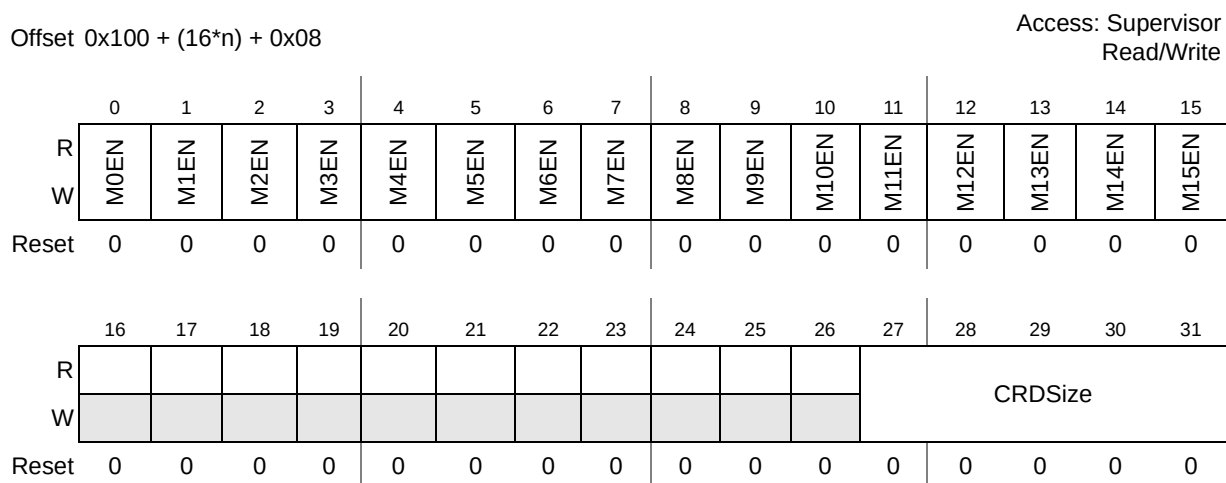
Figure 378. PFlash Calibration Region Descriptor n , Word2 (PFCRD n .Word2)

Table 454. PFlash Overlay Region Descriptor *n*, Word 2 description

Name	Description
0-15 MxEN	Calibration Remap Descriptor <i>n</i> Master <i>x</i> Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master <i>X</i> as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master <i>X</i> as defined by this region descriptor.
27-31 CRDSize	Calibration Remap Descriptor <i>n</i> Size - This field specifies the size of the calibration remap region. 00000 Region size is 1 byte. 00001 Region size is 2 bytes. Low-order bit of LSTARTADDR and PSTARTADDR must be zero 00010 Region size is 4 bytes. Low-order 2 bits of LSTARTADDR and PSTARTADDR must be zero 00011 Region size is 8 bytes. Low-order 3 bits of LSTARTADDR and PSTARTADDR must be zero 00100 Region size is 16 bytes. Low-order 4 bits of LSTARTADDR and PSTARTADDR must be zero 00101 Region size is 32 bytes. Low-order 5 bits of LSTARTADDR and PSTARTADDR must be zero 00110 Region size is 64 bytes. Low-order 6 bits of LSTARTADDR and PSTARTADDR must be zero 00111 Region size is 128 bytes. Low-order 7 bits of LSTARTADDR and PSTARTADDR must be zero 01000 Region size is 256 bytes. Low-order 8 bits of LSTARTADDR and PSTARTADDR must be zero 01001 Region size is 512 bytes. Low-order 9 bits of LSTARTADDR and PSTARTADDR must be zero 01010 Region size is 1 KB. Low-order 10 bits of LSTARTADDR and PSTARTADDR must be zero 01011 Region size is 2 KB. Low-order 11 bits of LSTARTADDR and PSTARTADDR must be zero 01100 Region size is 4 KB. Low-order 12 bits of LSTARTADDR and PSTARTADDR must be zero 01101 Region size is 8 KB. Low-order 13 bits of LSTARTADDR and PSTARTADDR must be zero 01110 Region size is 16 KB. Low-order 14 bits of LSTARTADDR and PSTARTADDR must be zero 01111 Region size is 32 KB. Low-order 15 bits of LSTARTADDR and PSTARTADDR must be zero 10000 Region size is 64 KB. Low-order 16 bits of LSTARTADDR and PSTARTADDR must be zero 10001 Region size is 128 KB. Low-order 17 bits of LSTARTADDR and PSTARTADDR must be zero 10010 Region size is 256 KB. Low-order 18 bits of LSTARTADDR and PSTARTADDR must be zero 10011 Region size is 512 KB. Low-order 19 bits of LSTARTADDR and PSTARTADDR must be zero 10100 Region size is 1 MB. Low-order 20 bits of LSTARTADDR and PSTARTADDR must be zero 10101 Region size is 2 MB. Low-order 21 bits of LSTARTADDR and PSTARTADDR must be zero 10110 Region size is 4 MB. Low-order 22 bits of LSTARTADDR and PSTARTADDR must be zero 10111 Region size is 8 MB. Low-order 23 bits of LSTARTADDR and PSTARTADDR must be zero 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

34.5 Functional description

As shown in [Figure 369](#) the flash memory controller interfaces between:

- AHB system bus ports
- ASC Modeling Unit (AMU)
- Nexus trace
- Flash memory array
- Overlay RAM
- Off-chip Buddy Device

For accesses targeting the flash array, the flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the flash array. The flash memory controller captures read data from the flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of two ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

34.5.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Thus, the access time of the operation is determined by the setting of the PFCR1[RWSC] field. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

34.5.2 Access protections

34.5.2.1 PFAPR - Platform Flash Access Protection Register (PFAPR)

The flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in [Section 34.4.1.4, "Platform Flash Access Protection Register \(PFAPR\)"](#). Detection of a protection violation results in an error response from the flash memory controller on the AHB transfer.

34.5.2.2 BAF execution disable

The flash memory controller provides a mechanism to disable instruction execution of the BAF (Boot Assist Flash) code. Setting the BAF Disable field (BAF_DIS) in the PFlash Configuration Register 3 (PFCR3) prevents execution of the BAF instructions. When this field is set, all incoming requests from the system bus on either Port0 or Port1 are evaluated. If the flash memory controller detects an attempted instruction fetch access to

the BAF region of the flash, the flash memory controller aborts the access and returns a system bus error. Data accesses to the BAF region are not affected by the state of PFCR3[BAF_DIS].

34.5.3 Read cycles - buffer miss

On an incoming AHB read request, a mini-cache lookup and access privilege evaluation are performed during the AHB address phase. In the event of a buffer miss, a flash access is initiated. If globally enabled, the calibration remap evaluation is also performed. If the access is to be remapped, the appropriate access is routed to the destination memory.

If the flash access was the direct result of an AHB transaction, the corresponding page buffer is loaded and marked as most-recently-used. If the flash access was the result of a speculative prefetch to the next sequential line, it is loaded into the least-recently-used buffer. The status of this buffer is not changed to most-recently-used until a subsequent buffer hit occurs as a result of an AHB read request.

34.5.4 Read cycles - buffer hit

Single cycle read responses to the AHB are possible with the flash memory controller when the requested read access was previously loaded into one of the page buffers. In these “buffer hit” cases, read data is returned on the system bus with a zero wait-state response.

34.5.5 Error termination

The flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding Read Access Control or Write Access Control settings do not allow the access, thus causing a protection violation. See ^(34.4.1.4) for more detail. In this case the flash memory controller does not initiate a flash array access.
- Attempted instruction fetch by an AHB master to the BAF flash region when the PFCR3[BAF_DISABLE] field in Platform Flash Configuration Register 3 (PFCR3) is set. The assertion of PFCR3[BAF_DISABLE] indicates that code stored in the BAF region should not be executed. See [Section 34.4.1.3, Platform Flash Configuration Register 3 \(PFCR3\)](#) for more detail. In this case, the flash memory controller does not initiate a flash array access.
- Attempted access by an AHB master to a flash region, where the corresponding censorship control is asserted.
- The flash or overlay RAM returns an error response on an attempted access (see ^(34.5.6)).
- Attempted access by an AHB master to a reserved region in the flash memory map.

34.5.6 Flash error response operation

The flash array may signal an error response to terminate a requested access due to improper sequencing during program/erase operations, improper sequencing during array integrity testing. When an error response is received, the flash memory controller does not update or validate a page read buffer. An error response may be signaled on read or interlock write operation. For more information on the specifics related to signaling of flash errors, including flash ECC events, array integrity testing and read-while-write events, refer to the flash NVM documentation.

34.5.7 Censorship

The entire flash space is defined by five major regions:

- Code flash (instruction and constant data)
- Data flash (data for EEPROM emulation):
- Secure code flash (instruction for security module)
- Secure data flash (data for security module)
- UTEST flash

Each section can be independently censored, and the flash memory controller provides independent censorship control inputs for each region on a read vs. write basis.

34.5.8 Security module exclusive control

During “Production at OEM” lifecycle state and beyond, the HSM can establish the secure flash blocks to be exclusive to the HSM and route program or erasing of secure flash blocks to the alternate programming interface.

The secure flash space is independently controlled for each of the following four regions:

- 16 KB secure code flash block at address range 0x0060_C0000 - 0x0060_FFFF
- 64 KB secure code flash blockA at address range 0x0061_0000 - 0x0061_FFFF
- 64 KB secure code flash blockB at address range 0x0062_0000 - 0x0062_FFFF
- Two 16 KB secure data flash blocks at address range 0x0068_0000 - 0x0068_7FFF

The HSM provides flags to individually control the HSM-exclusivity for each of the four secure flash regions. When a given secure flash block is marked as exclusive to the HSM, the HSM is the only system bus master granted access to that block and access attempts by non-HSM masters are terminated with error. Once HSM-exclusivity is established for a given secure flash block, the HSM is guaranteed access to that block, regardless of the state of the censorship controls. Secure flash exclusivity applies to read and write accesses.

The HSM provides flags to individually control the routing interlock write for each of the four secure flash regions to either the main program/erase interface or the alternate program/erase interface. When a given secure flash block is assigned to the alternate program interface, the flash controller appropriately routes incoming interlock write requests within the given secure flash block address range to the flash's alternate program interface. Alternate program control can be controlled independent of HSM exclusivity for a given secure flash block.

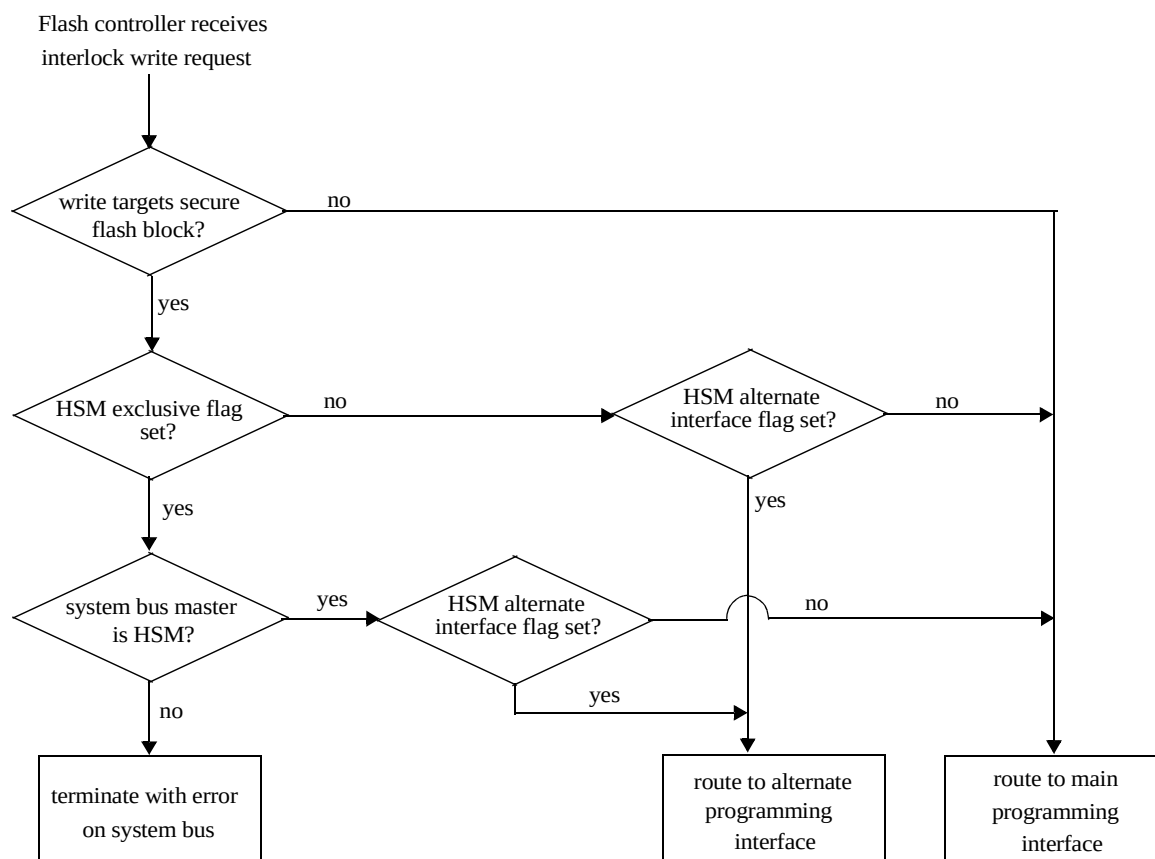


Figure 379. Interlock Write Alternate Programming Interface Routing Flowchart

34.5.9 Access pipelining

Accesses to the flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by the AHB master. Access pipelining may be applied only to read cycles targeting the flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal flash access and is not supported for calibration overlay RAM fetches. On calibration reads, the setting of PFCR1[APC] is ignored.

Note: Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.

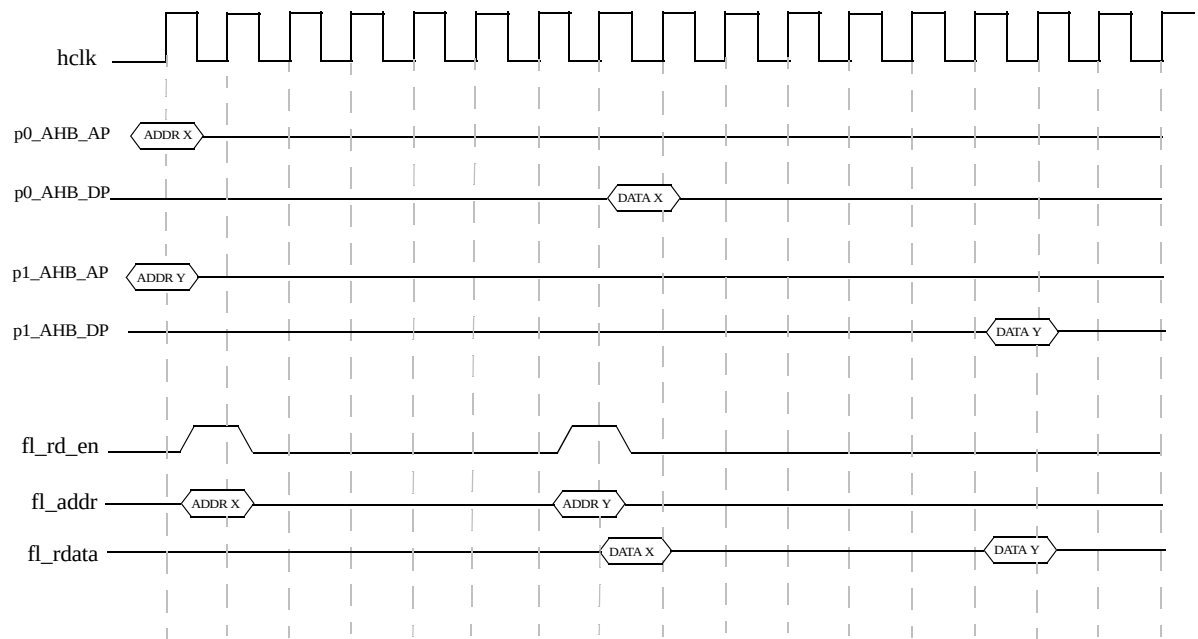


Figure 380. Port0 Access Followed By Port1 Access(RWSC=5, APC=0)

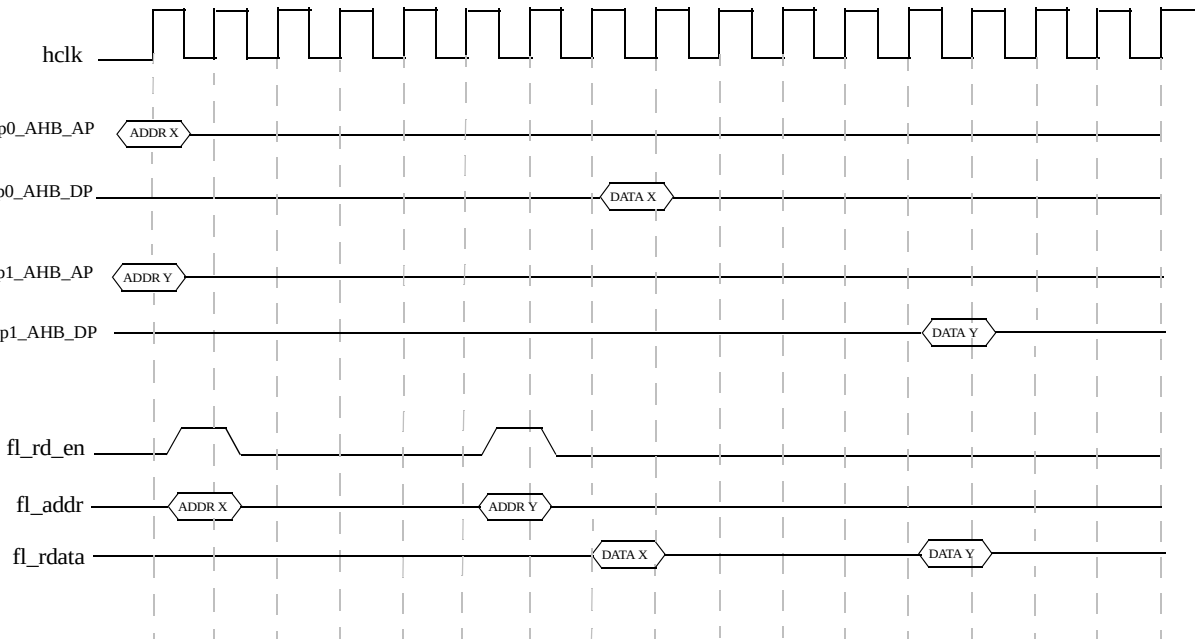


Figure 381. Port0 Access Followed By Port1 Access(RWSC=5, APC=1)

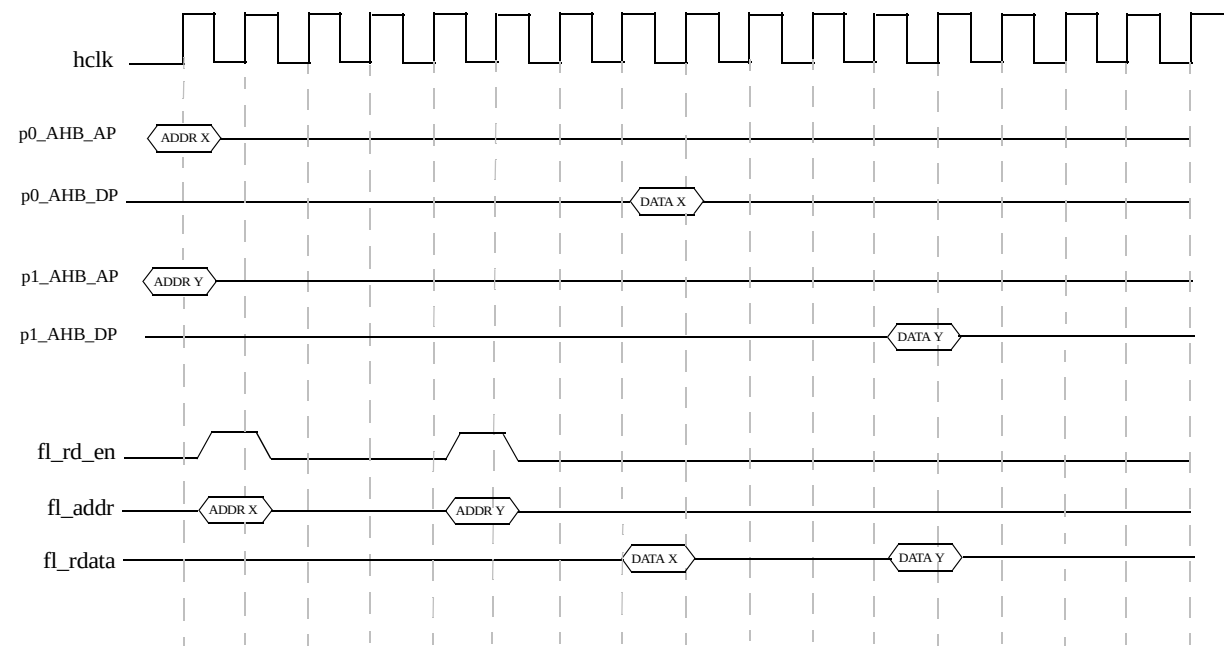


Figure 382. Port0 Access Followed By Port1 Access(RWSC=5, APC=2)

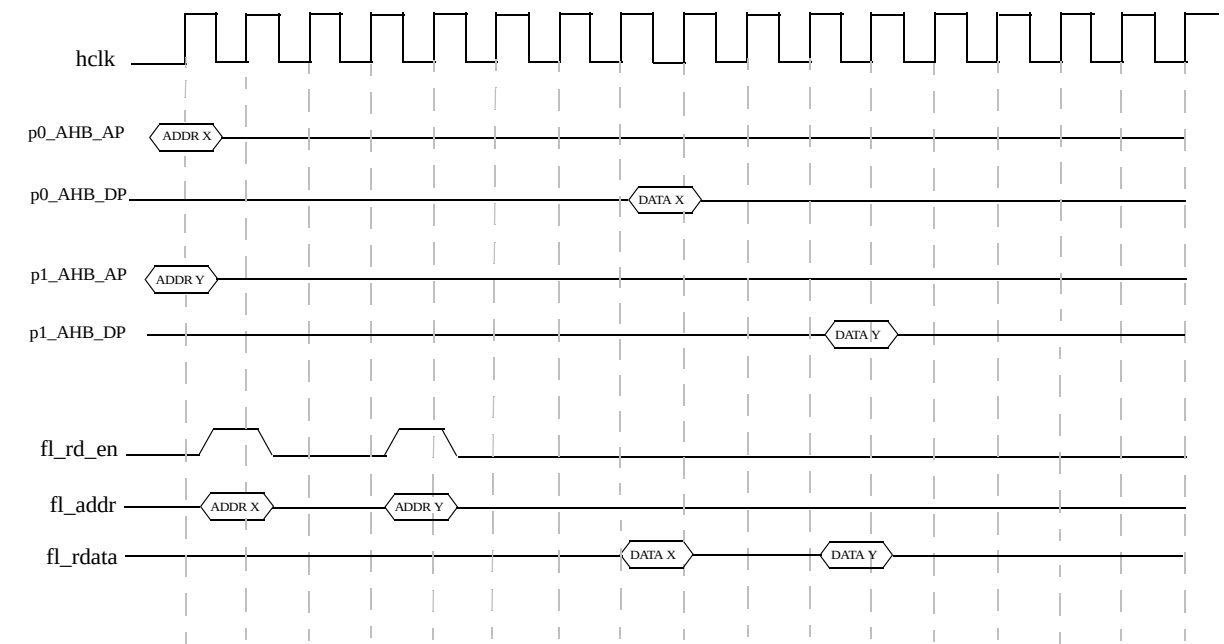


Figure 383. Port0 Access Followed By Port1 Access(RWSC=5, APC=3)

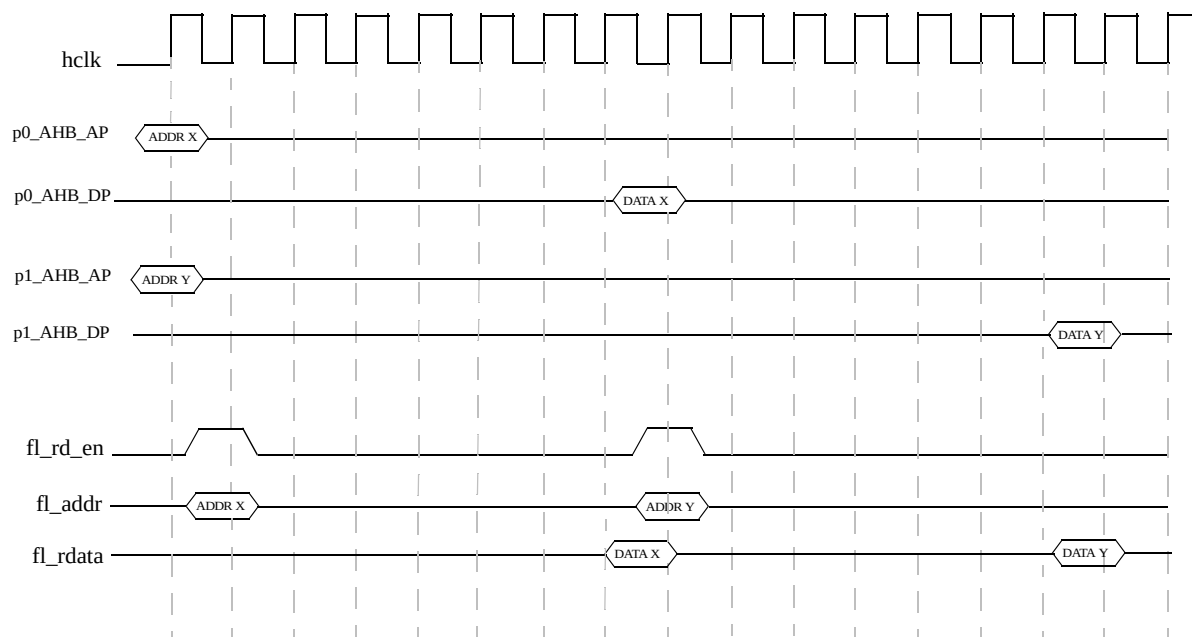


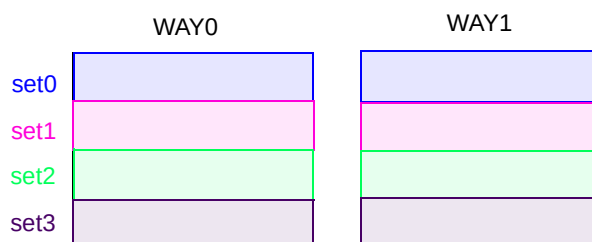
Figure 384. Port0 Access Followed By Port1 Access(RWSC=5, APC=3'b1xx)

34.5.10 Line read buffers and prefetch operation

The AHB ports of the flash memory controller each contain a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see [Section 34.4.1.1, "Platform Flash Configuration Register 1 \(PFCR1\)"](#)). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32-bytes, and a subsequent flash access is initiated. A flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

Figure 385. Flash memory controller 4-entry, 2-way mini-cache organization



Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in [Section 34.4.1.1, Platform Flash Configuration Register 1 \(PFCR1\)](#). More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR{1,2}[P{0,1}_BFEN] must be set to '1',
2. PFCR{1,2}[P{0,1}_PFLIM] must be non-zero, and
3. either PFCR{1,2}[P{0,1}_IPFEN] or PFCR{1,2}[P{0,1}_DPFEN] must be asserted.

Refer to [Section 34.4.1.1, Platform Flash Configuration Register 1 \(PFCR1\)](#) for a description of these controls.

34.5.11 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to [Section 34.4.1.1, Platform Flash Configuration Register 1 \(PFCR1\)](#) for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to [Section 34.4.1.2, Platform Flash Configuration Register 2 \(PFCR2\)](#) for a description of these controls. Prefetches are never triggered by write cycles.

34.5.12 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to [Section 34.4.1.1, Platform Flash Configuration Register 1 \(PFCR1\)](#) for a description of these controls.

34.5.13 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1}_WCFG) field. Refer to [Section 34.4.1.3, Platform Flash Configuration Register 3 \(PFCR3\)](#). The buffers can be organized as a “pool” of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, way 0 is allocated for instruction fetches and way 1 for data accesses.

34.5.14 PFlash calibration remap support

The flash memory controller supports calibration development by providing a remapping function to route flash accesses to overlay RAM. Recall this family of devices includes a comprehensive set of features including dedicated overlay RAM arrays to aid calibration and debug. The overlay function supports the following features:

- Can be mapped over internal flash memory
 - Allows calibration of constant data without requirement for additional external RAMs and calibration memory interfaces
- Can be used to hold device debug trace stream
 - Allows limited trace based debug without requirement for complex debug tool hardware and access to high speed trace port
- Protected with error detection and correction mechanism
 - Accesses to all overlay RAMs can achieve the same timing as accesses to internal flash. Timing for calibration accesses to a particular overlay RAM is only guaranteed if it is not being used as a destination for trace streaming at the same time
 - Timing for calibration accesses to overlay RAM is only guaranteed if RWSC is set sufficiently high to cover the intrinsic burst access incurred when fetching a full page of calibration data
- Support for up to 32 distinct calibration remap regions.
- Protection against accidental enable and reconfiguration of calibration remap function:
 - Two stage mechanism to enable remap, using both global and individual calibration region descriptor enable bits
 - Optional configuration of calibration region descriptors as redundant pairs, allowing detection of accidental region reconfiguration.
- Support for overlay remapping to unique calibration regions on a per-master basis.

The calibration remap function supports the following overlay RAM targets:

- An internal overlay RAM is included on all standard production devices.
- An extended overlay RAM is an additional memory that is only available on special ED devices.
- A portion of the system RAM can be used for overlay.

The overlay RAM is selected based on the translated physical address.

To better understand the functionality of the calibration remap, consider more detailed block diagrams, one showing the overall structure of the calibration remap function and another showing more detail on the actual remap logic. Recall the flash memory controller implements a hardware interconnection matrix connecting the four input ports and four destination memories. See [Figure 386](#).

The following figure illustrates the calibration remap connection matrix.

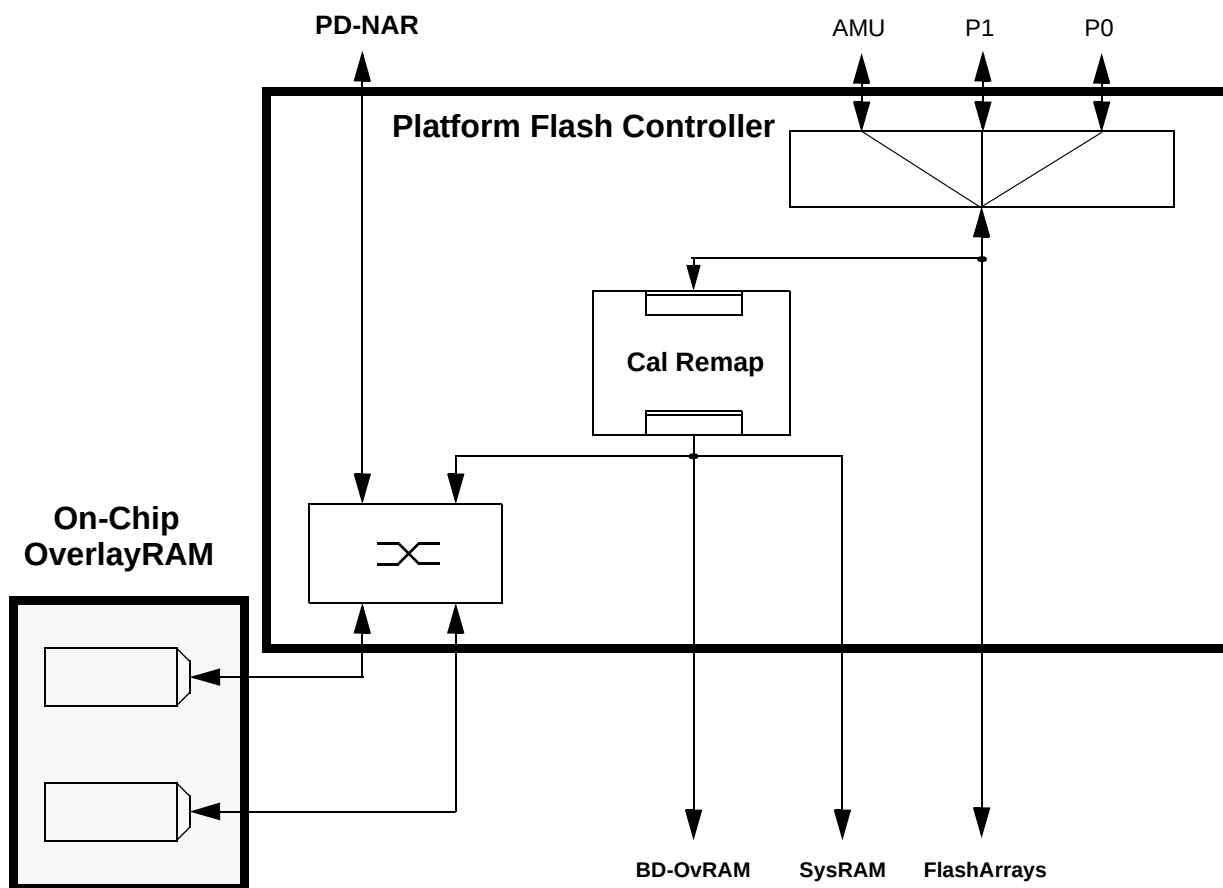


Figure 386. PFlash calibration remap connection matrix

Note the 64-bit on-chip overlay RAM is split into 2 equal-sized memories to allow concurrent accesses for calibration data from the P0, P1 (where these are the crossbar slave input ports) and AMU ports plus trace data writes from the Production Device Nexus Aurora Router (PD-NAR).

The translated physical address defines the destination memory. [Table 455](#) shows the base addresses for the calibration data memories as defined by the system memory map.

Table 455. Calibration data memory base addresses

Calibration Memory	Base Address	Possible Access Sources
Flash Arrays	0x0{8,9,A}-_----	P0, P1, AMU
System RAM	0x40{0,1}-_----	P0, P1, AMU
On-chip Overlay RAM	0x0D00_----	P0, P1, AMU, PD-NAR
Buddy Device (Extended) Overlay RAM	0x0C{0,1}-_----	P0, P1, AMU

Note the trace data from the Nexus Aurora Router (PD-NAR) can only be routed to the on-chip overlay RAM; there is a different connection path in the Buddy Device from the BD-NAR into the extended overlay RAM.

The system RAM connection from the calibration remap logic is implemented via a *private 64-bit connection to the platform RAM controller*.

Accesses to the flash arrays always use the logical, non-remapped system address. Calibration data store references to the system RAM, on-chip or buddy device (extended) overlay RAM always use the physical, non-remapped system address. Conversely, calibration data load references can be accessed using a *translated physical address or, if provided by the requesting master, the physical, non-remapped system address*. Byte-, halfword-, word- and doubleword-sized data store references to the on-chip overlay RAM are supported. Byte-, halfword-, word- and doubleword-sized store references to the buddy device (extended) overlay RAM are supported.

The read data interface of system RAM, on-chip overlay RAM and buddy (extended) device is 64 bits wide for each, whereas the flash read data interface is 256 bits wide. As a result, a remapped calibration load access initiates a four-beat burst sequence to mimic the throughput of a single flash access. There is a direct relationship between the operating frequency range and the required number of wait states to correctly sample flash read data. By contrast, a remapped calibration load access has a fixed, intrinsic 4-beat latency, where the access time of each individual beat depends on the overlay target - system RAM, on-chip overlay RAM or buddy device. The access time when fetching data from the buddy device presents the longest latency. Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. Expressed mathematically:

$$(RWSC + 2) \geq \left(\frac{\text{IntrinsicFlashAccessTime}}{\text{OperatingFrequencyClockPeriod}} + 2 \right) \geq (\text{OverlayRAMTransferBurstLatency})$$

Notice the overall latency on a random, initial flash access that does not hit in the prefetch mini-buffer incurs an additional cycle latency beyond the intrinsic flash access time. This cycle accounts for pipeline stalls incurred by the flash controller state machine. The formula above is based on specified intrinsic flash access times of the C55FMC array at 150C. Calibration data that is loaded in the flash controller's mini-cache is returned with single-cycle latency.

The memory-mapped calibration remap (region) descriptor (CRDn) registers define the required address translation. Each calibration remap descriptor includes the logical base address, the translated physical base address and a region size (plus other control bits). The calibration remap evaluation is initiated when the flash controller is presented with a flash access request from any system bus master to a location in the mirrored flash address space. Conversely, references using the associated non-mirrored flash system address are not evaluated for calibration remapping. Generally, not all areas of the flash memory map are available for calibration remapping. Secure flash blocks, EEPROM, BAF and UTEST are usually not available for calibration remapping. Flash access requests which present an address targeting the secure flash blocks, EEPROM, BAF or UTEST will bypass calibration remap evaluation and return flash data.

The address translation mechanism requires the logical and physical base addresses of the region be aligned on 0-modulo-size boundaries. See [Figure 387](#).

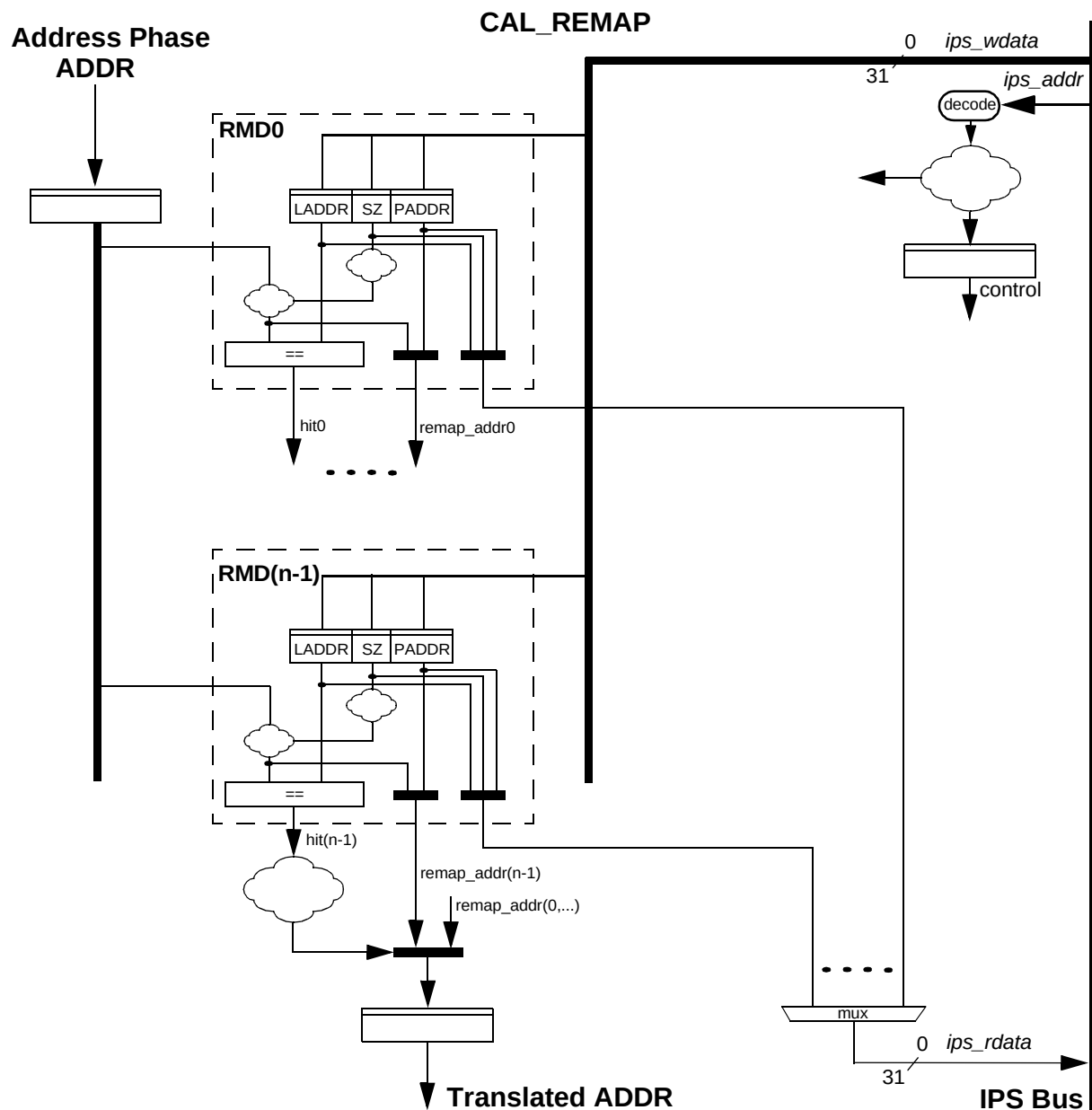


Figure 387. PFlash calibration remap logic detail

Each instantiation of the basic calibration remap function $\{RMD0, \dots, RMD(n-1)\}$ in [Figure 387](#) includes the three descriptor registers containing the logical start address (LADDR in the figure), the physical start address (PADDR) and the region size (SZ). The region size is decoded to create the appropriate address bit enables which are then applied to the registered logical flash address. An equality comparator then compares the adjusted logical flash address against the logical address from the calibration remap descriptor to

determine the region “hit”. The hit determination is further qualified based on the bus master number that initiated the flash access. At the same time, the logical flash address and the physical address defined in the calibration remap descriptor are merged to form the translated address to be used in the event of a calibration region hit. Aside from the individual instantiations of the calibration remap descriptors, this logic examines all the descriptor hit indicators to select the appropriate translated address.

In the absence of a region “hit” data is returned from the flash at the provided logical address.

It should be noted in the event of *overlapping remap regions*, the calibration remap descriptor with the largest number is used for the address translation, that is, if a logical flash address hits in multiple descriptors, say CRD_x and CRD_y, the translated address from CRD_y is used, given $y > x$. This definition is selected as it minimizes the gate cost and timing impact of the calibration remap descriptor “hit” logic.

In general, the calibration remap logic operates only on flash *data* accesses. However, flash instruction references can optionally remapped while in debug modes of operation to support the use of software breakpoints when PFCRC[IRMEN] is set.

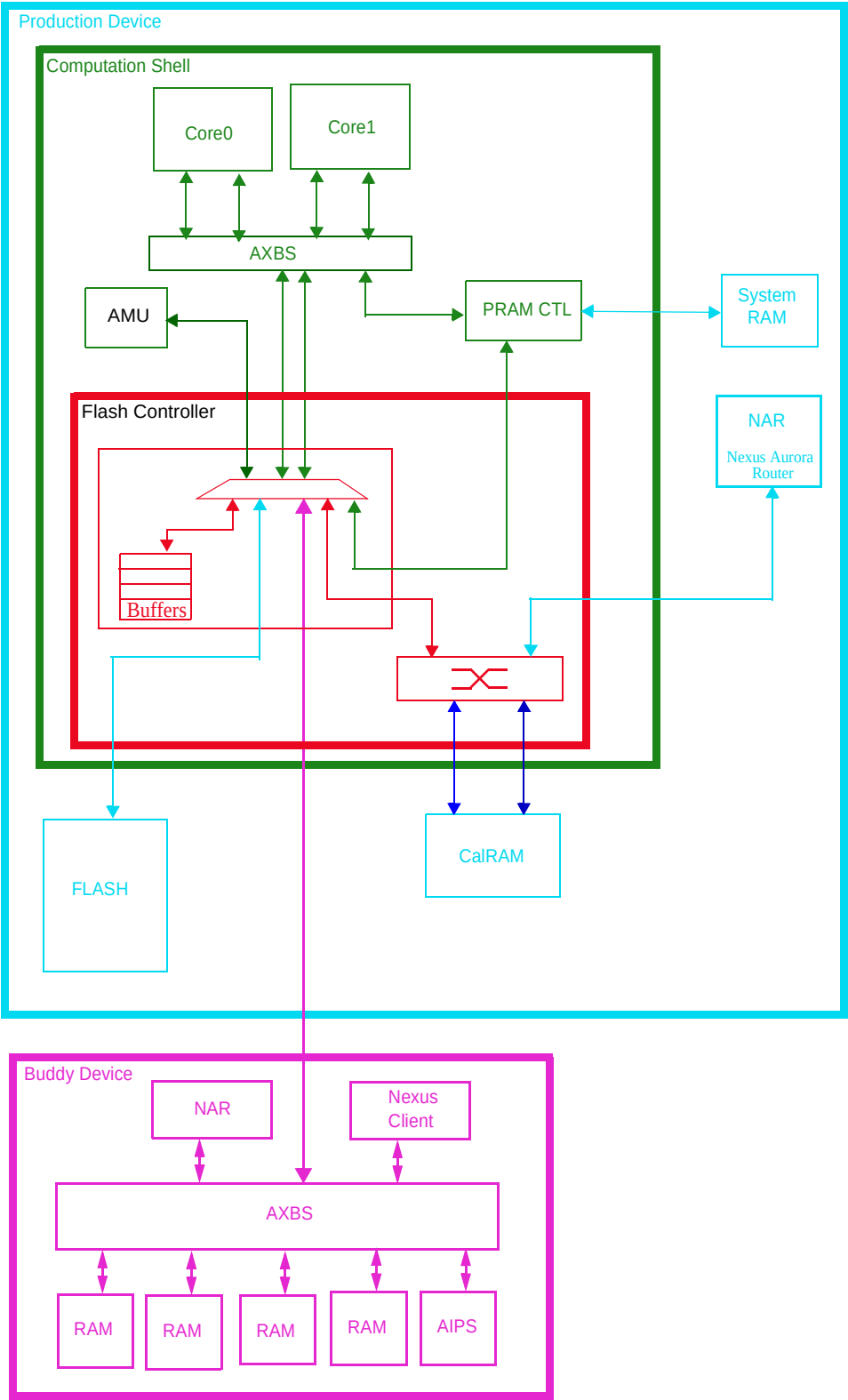


Figure 388. Flash memory controller and associated calibration remap targets

34.5.14.1 Overlay Access and Buddy Device Considerations

34.5.15 AMU dedicated flash port

The flash memory controller provides a dedicated flash interface for servicing the DMA function in the ASC Modeling unit (AMU). This facilitates the AMU's minimum bandwidth requirement for feeding the dual calculation channels, which process complex multidimensional data models using a kind of neural network technology. The AMU uses the DMA to move data tables from the flash to a local RAM. In the event of concurrent flash requests from the crossbar slave port(s) of the flash controller and the AMU, the crossbar slave port(s) have priority over the AMU by default. However, the AMU can dynamically elevate its priority over the crossbar slave ports to avoid starvation of its computation engine.

34.5.16 Safety considerations

34.5.16.1 e2eECC (End-to-End ECC)

34.5.16.1.1 e2eECC and flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with flash memory. The basic issues involve: the default state of a flash block that is erased (all ones) and the fact that programming an erased flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the flash memory in three ways.

First, the all ones codeword, since it reflects the state of an erased flash location, is treated as a valid, error free encoding; in particular, it is required that the checkbits associated with the all ones data value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased flash block generates an all ones data for all locations, the address field cannot be included in the ECC code used by the flash. The flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the flash memory and its support for EEPROM emulation. Lastly, there are ECC implications associated with the storage and retrieval of overlaid calibration data.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the flash memory in multiple ways. The required ECC adjustments are handled by the platform flash controller before data is written to the flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all ones erased state.

On a flash read operation, a similar but "reversed" set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following flash ECC example. Let the flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns `fl_rdata = 0xFFFFFFFF_FFFFFFFF`, `fl_cdata = 0xFF`. The flash memory controller calculates the `addr_chkbit = 0xC4` using the H-matrix; it inverts the `fl_cdata` vector to 0x00 and factors in the `addr_chkbit = 0xC4` to produce the following valid e2eECC codeword:

`addr = 0x00345678`, `rdata = 0xFFFFFFFF_FFFFFFFF`, `rchkbit = 0xC4` ($0x00 \wedge 0xC4$)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

`addr = 0x00345678`, `wdata = 0xBABEFACE_DEADBEEF`, `wchkbit = 0xE3`

Before performing the interlock write to the flash array, the flash memory controller adjusts the `chkbit` value as `fl_wchkbit = 0xD8` ($0xE3 \wedge 0xFF \wedge 0xC4$) by toggling the original write checkbits and then factoring in the `addr_ecc`. Accordingly, the codeword stored in the flash array is:

`fl_data = 0xBABEFACE_DEADBEEF`, `fl_checkbit = 0xD8`

3. Read address 0x00345678. The flash array returns the stored codeword, `fl_rdata = 0xBABEFACE_DEADBEEF`, `fl_rchkbit = 0xD8`. The flash memory controller inverts the checkbit vector and factors in the address checkbits ($0xD8 \wedge 0xFF \wedge 0xC4$) to produce:

`addr = 0x00345678`, `rdata = 0xBABEFACE_DEADBEEF`, `rchkbit = 0xE3`

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The flash memory controller performs EDC after ECC check on all flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

34.5.16.2 Flash address generation check

Functional safety coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Functional safety coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. Recall on a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash interface signals to invoke a flash lookup. In addition to providing the requested read data, the flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.

This sideband information is used by the flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

34.5.16.2.1 On-Chip Overlay RAM Feedback Check

The integrity of the address and data on an overlay RAM transaction is covered by e2eECC check performed by the requesting master. Functional safety coverage of the address path and control within the flash memory controller is handled by a transaction monitor which verifies the integrity of the transactions between the flash memory controller and the on-chip overlay RAM. The function of the transaction monitor relies on a feedback path between the

flash memory controller and the on-chip overlay RAM, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated. This feedback information is used by the flash memory controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and on-chip overlay RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

34.5.16.2.2 e2eECC and Calibration Accesses

Because calibration write accesses are always presented with the physical, non-remapped system address, the write data checkbits presented by the master are calculated based on the physical overlay RAM system address. On a subsequent overlay read, the ECC checkbits must be manipulated to replace the physical overlay RAM address contribution with the logical flash address contribution. This is required to satisfy the e2eECC check at the originating master.

In addition, the flash memory controller can be configured to treat calibration RAM space as a safety-critical component. By programming `PFCRCR[SAFE_CAL]=1`, the flash memory controller treats the 32 available calibration region descriptors (CRD0-CRD31) as a replicated group of 16 region descriptors. More specifically, when N is the total number of region descriptors:

- The contents of CRD0 are expected to be redundantly defined in CRD[N/2]
- The contents of CRD1 are expected to be redundantly defined in CRD[(N/2)+1]
- The contents of CRD2 are expected to be redundantly defined in CRD[(N/2)+2]
- ...and so on
- The contents of CRD[(N/2)-1] are expected to be redundantly defined in CRD[N-1]

Note: It is the user's responsibility to establish redundant calibration region descriptors by programming the associated pairs of CRDs.

Incoming flash read requests are compared simultaneously against the calibration regions defined in CRD0 to CRD[(N/2)-1] and the calibration regions defined in CRD[N/2] to CRD[N-1]. Due to the enabled redundancy, the parallel overlay remap hardware structures should evaluate to symmetric results. If a mismatch is detected, the event is reported to the FCCU and calibration remapping is not performed. Instead, data is returned from the flash.

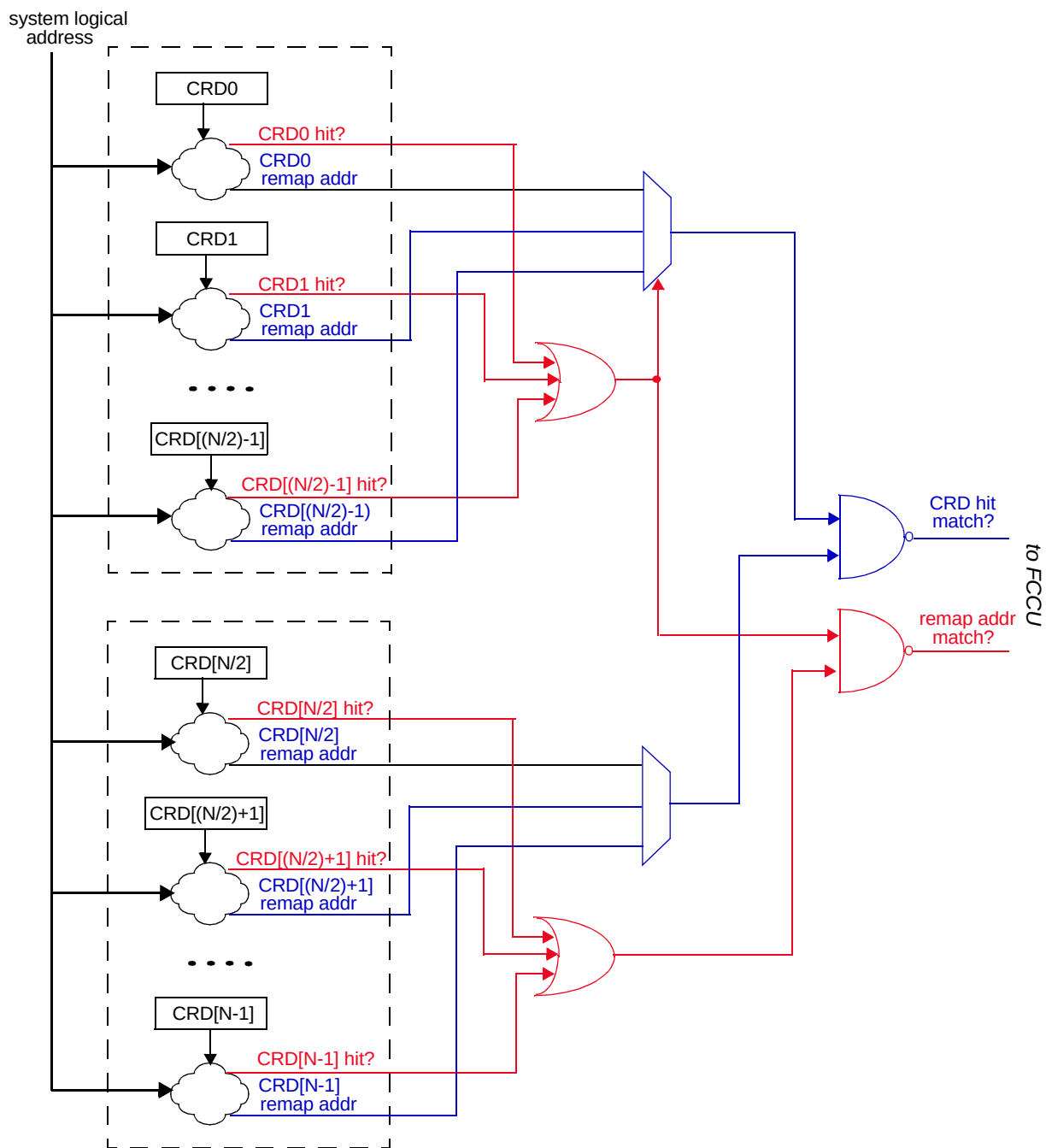


Figure 389. Safety-critical calibration remap datapath with redundancy

34.5.17 e2eECC and EEPROM accesses

ECC events detected on accesses to the EEPROM flash blocks are suppressed from being reported to the Memory Error Management Unit (MEMU).

In the event of a single-bit correction, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is suppressed from being reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode value is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address, and the non-correctable error event is suppressed from being reported to the MEMU.

34.5.18 Array integrity considerations

During an array integrity sequence, the flash memory array ignores any incoming read requests. When a flash array integrity check is in progress, the flash memory controller terminates all flash access requests with an error.

35 Embedded Flash Memory (c55fmc)

35.1 Introduction

The primary function of the embedded flash memory is to serve as electrically programmable and erasable non-volatile memory (NVM) that may be used for instruction or data storage.

Figure 390 shows the top-level diagram and functional organization of the flash memory unit.

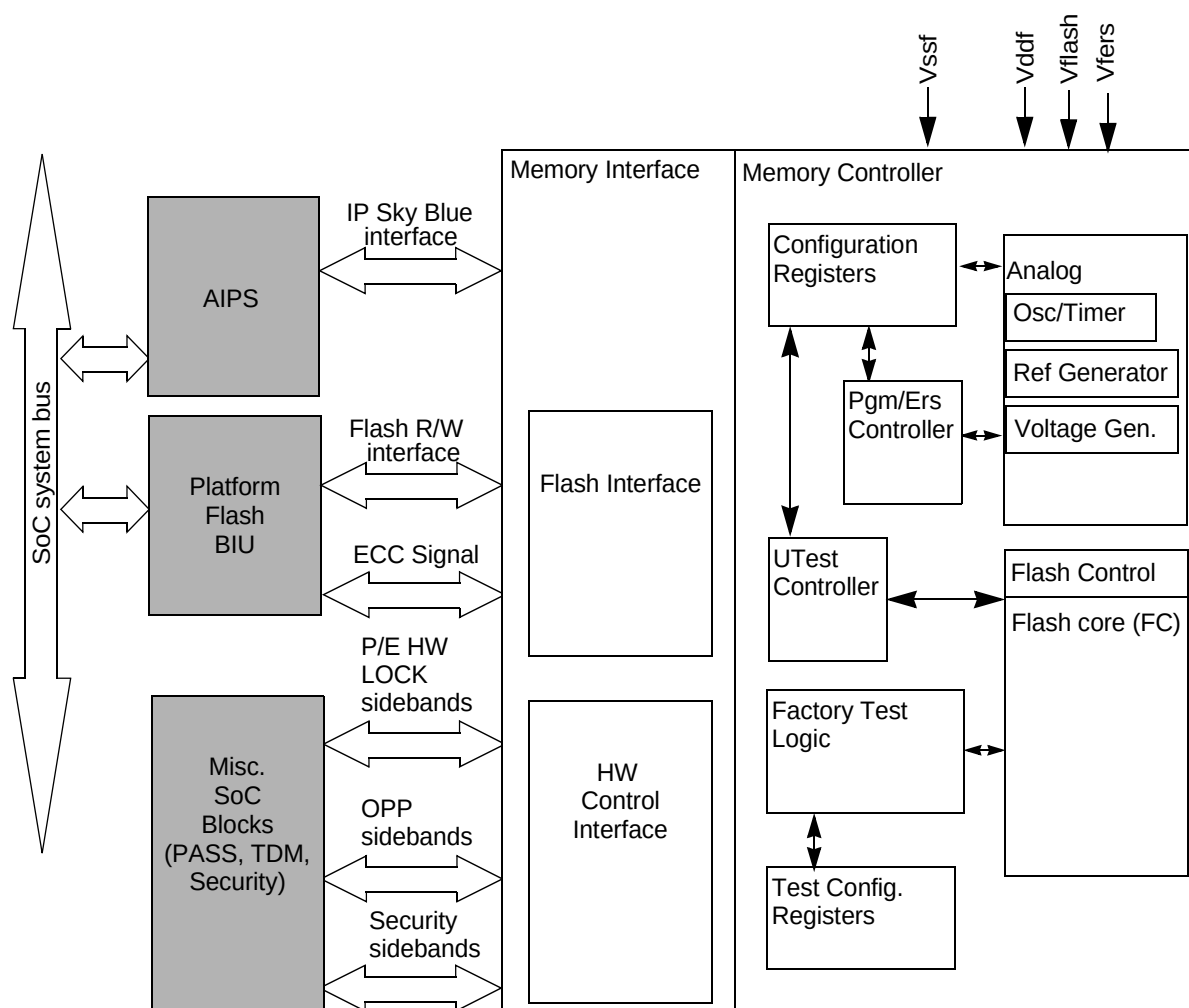


Figure 390. Block diagram of c55fmc

35.1.1 Overview

The embedded flash memory is addressable by word (32 bits) or double word (64 bits) for program operation, and page (256 bits) for read operation. Multiple word or double-word writes may be done to the flash memory to fill up the program page buffer (256 bits),

enabling page programming (256 bits, requiring 4 double-word writes) and quad-page programming (1024 bits, requiring 16 double-word writes). Flash memory reads always return 256 bits, although read page buffering may be performed by the Bus Interface Unit (BIU).

For more details on the embedded flash memory architecture and features, see [Section 35.3, “Functional Description”](#).

35.1.2 Features

The embedded flash memory includes these distinct features:

- Test information stored in a dedicated nonvolatile block (referred to as the UTest block)
- OTP space made available in the UTest block
- Read page size of 256 bits (8 words)
- ECC with single-bit correction, double-bit detection (all 1's valid)
- Quad Page programming (64-bit granularity)
- Software programmable block program/erase restriction control
- Erasing of selected block(s)
- Independent programming of the UTest NVM block
- Embedded hardware program and erase algorithms
- Support for reading while writing when the accesses are to different partitions
- Erase suspend, program suspend, and erase-suspended program
- UTest mode (user-accessible test modes) including Array Integrity and Margin Read
- Triple Voted Flops for safety critical flash functions

35.1.3 Modes of operation

Here is a brief description of the embedded flash memory operating modes.

- *User mode* is the default operating mode of the embedded flash memory. In this mode, it is possible to read and write registers, read and interlock write the memory array, program the memory array, and erase the memory array. In this mode program and erase operations are initiated by doing array and register writes, and are controlled by an internal state machine.
- *Low power mode* turns off all DC current sources within the embedded flash memory on the Vflash domain, and enables VDDF to be power gated. The embedded flash memory is not accessible for read, write, program, or erase when in low power mode. This is the lowest current draw the embedded flash memory can achieve.
- *UTest mode* is a tiered test mode strategy in which a portion of the factory test modes are made available. This mode is protected but accessible.

35.2 Memory map and register definition

The embedded flash memory map consists of a flash memory array (which includes main array space and UTest NVM space) as well as a second region of registers associated with the programming model that enable flash memory array operation and modification.

35.2.1 Memory map

The address space consists of 16 KB, 32 KB, 64 KB, and 256 KB blocks.

Table 456. Block Memory Map

Offset or address	Memory or register	Access	Reset value	Section/page
0x000	MCR—Module Configuration Register	R/W ⁽¹⁾	0x0000_0200	35.2.3.1/1083
0x004 ⁽²⁾	MCRA—Alternate Module Configuration Register	R/W ⁽³⁾	0x0000_0200	35.2.3.2/1089
0x004	Reserved	R	0x0000_0000	—
0x008	MCRE—Extended Module Configuration Register	R/W ⁽⁴⁾	0x####_#### ⁽⁵⁾	35.2.3.3/1090
0x00C	Reserved	R	0x0000_0000	—
0x010	LOCK0—Lock 0 register	R/W	0xBFFF_FFFF	35.2.3.4/1093
0x014	LOCK1—Lock 1 register	R/W	0x0000_FFFF	35.2.3.5/1097
0x018	LOCK2—Lock 2 register	R/W	0xFFFF_FFFF	35.2.3.6/1099
0x01C	LOCK3—Lock 3 register	R/W	0x0000_FFFF	35.2.3.7/1100
0x020	Reserved	R	0x0000_0000	—
0x024	Reserved	R	0x0000_0000	—
0x028 ⁽⁶⁾	LOCK0A—Alternate Lock 0 register	R/W	0xBFFF_FFFF	35.2.3.8/1101
0x028	Reserved	R	0x0000_0000	—
0x02C ⁽⁷⁾	LOCK1A—Alternate Lock 1 register	R/W	0x0000_FFFF	35.2.3.5/1097
0x02C	Reserved	R	0x0000_0000	—
0x030	Reserved	R	0x0000_0000	—
0x034	Reserved	R	0x0000_0000	—
0x038	SEL0—Select 0 register	R/W	0x0000_0000	35.2.3.10/1106
0x03C	SEL1—Select 1 register	R/W	0x0000_0000	35.2.3.11/1107
0x040	SEL2—Select 2 register	R/W	0x0000_0000	35.2.3.12/1108
0x044	SEL3—Select 3 register	R/W	0x0000_0000	35.2.3.13/1109
0x048	Reserved	R	0x0000_0000	—
0x04C	Reserved	R	0x0000_0000	—
0x050	ADR—Address register	R	0x0000_0000	35.2.3.14/1110
0x054	UT0—UTest 0 register	R/W	0x0000_0001	35.2.3.15/1112
0x058	UM0—UMISR 0 register	R/W	0x0000_0000	35.2.3.16/1115
0x05C	UM1—UMISR 1 register	R/W	0x0000_0000	35.2.3.16/1115
0x060	UM2—UMISR 2 register	R/W	0x0000_0000	35.2.3.16/1115
0x064	UM3—UMISR 3 register	R/W	0x0000_0000	35.2.3.16/1115

Table 456. Block Memory Map (continued)

Offset or address	Memory or register	Access	Reset value	Section/page
0x068	UM4—UMISR 4 register	R/W	0x0000_0000	35.2.3.16/1115
0x06C	UM5—UMISR 5 register	R/W	0x0000_0000	35.2.3.16/1115
0x070	UM6—UMISR 6 register	R/W	0x0000_0000	35.2.3.16/1115
0x074	UM7—UMISR 7 register	R/W	0x0000_0000	35.2.3.16/1115
0x078	UM8—UMISR 8 register	R/W	0x0000_0000	35.2.3.16/1115
0x07C	UM9—UMISR 9 register	R/W	0x0000_0000	35.2.3.16/1115
0x080	OPP0—Over-Program Protection 0 register	R	0x####_#### ⁽⁸⁾	35.2.3.17/1125
0x084	OPP1—Over-Program Protection 1 register	R	0x####_#### ⁽⁹⁾	35.2.3.18/1127
0x088	OPP2—Over-Program Protection 2 register	R	0x####_#### ⁽¹⁰⁾	35.2.3.19/1128
0x08C	OPP3—Over-Program Protection 3 register	R	0x####_#### ⁽¹¹⁾	35.2.3.20/1129
0x090	TMD—Test Mode Disable Password Check register	R/W	0x0000_0000	35.2.3.21/1130
0x94 to Max ⁽¹²⁾	Factory use only — Restricted Use	R	0x0000_0000	Restricted

1. Note that R/W registers may contain some read-only or write-only bits.
2. Alternate Interface registers may be re-mapped to a different base address.
3. Note that R/W registers may contain some read-only or write-only bits.
4. Note that R/W registers may contain some read-only or write-only bits.
5. # stands for configuration-dependent reset value.
6. Alternate Interface registers may be re-mapped to a different base address.
7. Alternate Interface registers may be re-mapped to a different base address.
8. # stands for configuration-dependent reset value.
9. # stands for configuration-dependent reset value.
10. # stands for configuration-dependent reset value.
11. # stands for configuration-dependent reset value.
12. Maximum address for flash register space is 0x1FC.

35.2.1.1 UTest NVM block

The UTest NVM block may be enabled by the BIU — this is device-specific. When the UTest NVM space is enabled, all operations are mapped to the UTest NVM block. User-mode programming of the UTest NVM block is enabled only when MCR[PEAS] is high. The UTest NVM block is an OTP block (assuming Test Mode Disable Seal is written) — thus erase is not allowed.

The UTest NVM block supports RWW, and is grouped with the blocks in partition 0.

The UTest NVM block may be locked against program by using the lock registers.

Programming of the UTest NVM space has restrictions that are similar to those of the main space in terms of how ECC is calculated. Only one program operation is allowed per 64-bit ECC segment.

The UTest NVM block contains specified data that is needed for embedded flash memory or SoC features.

35.2.1.2 Test Mode Disable Seal

Also included in the UTest NVM block is a mechanism to disable factory entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method will not be able to have possible failures analyzed by factory failure analysts. Once sealed, the Utest block becomes OTP (Erase Locked, and Over Program Protection enabled). The Utest Block also is not accessible in Test Mode.

Protection of this sort prevents all high-voltage operations to the flash executed by the internal state-machine, as well as reads through this state machine, and reads through the Array Integrity state machine when using Test mode interfaces.

Utest operations Margin Read and Array Integrity are also protected by preventing MISR updates on blocks selected for protection, although reads will still be done and Single Bit Corrections and Double Bit Detections will be logged. Protection through other interfaces is protected by normal User mode protection mechanisms, and are device-specific.

The method to disable factory entry into Test mode is to first program the Test Mode Disable Seal location to be 0x2D3C_4B5A. Once the next reset is asserted, Test mode will be disabled.

It is possible to create a password to enable factory entry into test mode. This can be programmed into the Test Mode Disable Override Passcode. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and will not be accepted to override. If it is desired by the customer that override never be possible, one of the three invalid passcodes should be put into this location. Passcode may be entered to authenticate entry (if enabled) by doing a 32-bit register write to register address 0x90. The UTest NVM block is always protected even if the Test Mode Disable Seal password is written.

Only blocks selected in the Test Mode Disable Block Select field are controlled by the Test Mode Disable feature. Thus it is possible for customers to selectively pick blocks that have this type of protection, and will not be eligible for factory failure analysis. Bits programmed to 0 in the Test Mode Disable Block Select field(s) will designate blocks that are controlled by the Test Mode Disable feature. If either Test Mode Disable Block Select - 0 or Test Mode Disable Block Select - 1 have a 0 programmed, that block will be protected. In order to let two opportunities to select blocks for the Test Mode Disable, two regions are available, Test Mode Disable Block Select - 0 and Test Mode Disable Block Select - 1. The bit of these two regions are logically ANDed, to define which blocks are effectively selected for the Test Mode Disable feature. The Block Select field is organized as follows, and aligns with how blocks are defined in the LOCK registers.

Table 457. Test mode disable block select

Block	Bits used to select blocks
Low Block Disable	Data[13:0]
Unused	Data[15:14]

Table 457. Test mode disable block select (continued)

Block	Bits used to select blocks
Mid Block Disable	Data[31:16]
High Block Disable	Data[47:32]
256K Block Disable	Data[95:48]

35.2.2 Register Summary

[Table 458](#) provides a key for register figures and tables.

Table 458. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register field types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero (previously designated slfclr).
Reset Values	
0	Resets to zero.
1	Resets to one.
1*	Resets to one based on UTest NVM block being erased. If UTest NVM block is not erased, than value is determined by value programmed in UTest NVM block.
—	Undefined at reset.
u	Unaffected by reset.
[signal_name]	Reset value is determined by polarity of indicated signal.

[Table 459](#) is the register summary table for embedded flash memory. These registers are associated with the programming model that enables flash memory array operation and modification. Reset values for these registers can be found in [Table 456](#).

Table 459. Register summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 MCR	R	RVE	RRE	AEE	EEE	0	0	0	0	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c												
	R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	FERS	0	0	PGM	PSUS	ERS	ESUS	EHV
	W	w1c	w1c	w1c					PECIE	FERS			PGM	PSUS	ERS	ESUS	EHV
0x004 MCRa	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PEASa	DONEa	PEGa	0	0	0	0	PGMa	PSUSa	ERSa	ESUSa	EHVa
	W												PGMa	PSUSa	ERSa	ESUSa	EHVa
0x008 MCRE	R	HT	0	0	n256K					n64Kh			n32Kh		n16Kh		
	W																
	R	n64Km			n32Km		n16Km			n64Kl			n32Kl		n16Kl		
	W																
0x010 LOCK0	R	TSLOCK	0	LOWLOCK													
	W																
	R	MIDLOCK															
	W																
0x014 LOCK1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	HIGHLOCK															
	W																
0x018 LOCK2	R	256KLOCK															
	W																
	R	256KLOCK															
	W																
0x01C LOCK3	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	256KLOCK															
	W																

Table 459. Register summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028 LOCK0a	R	TSLOCKa	0	LOWLOCKa													
	W																
	R	MIDLOCKa															
	W																
0x02C LOCK1a	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	HIGHLOCKa															
	W																
0x038 SEL0	R	0	0	LOWSEL													
	W																
	R	MIDSEL															
	W																
0x03C SEL1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	HIGHSEL															
	W																
0x040 SEL2	R	256KSEL															
	W																
	R	256KSEL															
	W																
0x044 SEL3	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	256KSEL															
	W																
0x050 ADR	R	SAD	aH	aM	aL	a256k	a64k	a32k	a16k	ADDR							
	W																
	R	ADDR													0	0	0
	W																
0x054 UT0	R	UTE	SBCE	0	0	0	0	0	0	0	0	0	0	0	CPR	CPA	CPE
	W																
	R	0	0	0	0	0	0	NAIBP	AIBPE	0	AISUS	MRE	MRV	0	AIS	AIE	AID
	W							NAIBP	AIBPE		AISUS	MRE	MRV		AIS	AIE	AID

Table 459. Register summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058 UM0	R	MISR															
	W																
	R	MISR															
	W																
0x05C UM1	R	MISR															
	W																
	R	MISR															
	W																
0x060 UM2	R	MISR															
	W																
	R	MISR															
	W																
0x064 UM3	R	MISR															
	W																
	R	MISR															
	W																
0x068 UM4	R	MISR															
	W																
	R	MISR															
	W																
0x06C UM5	R	MISR															
	W																
	R	MISR															
	W																
0x070 UM6	R	MISR															
	W																
	R	MISR															
	W																
0x074 UM7	R	MISR															
	W																
	R	MISR															
	W																

Table 459. Register summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x078 UM8	R	MISR															
	W																
	R	MISR															
	W																
0x07C UM9	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MISR
	W																
0x080 OPP0	R	0	0	LOWOPP													
	W																
	R	MIDOPP															
	W																
0x084 OPP1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	HIGHOPP															
	W																
0x088 OPP2	R	256KOPP															
	W																
	R	256KOPP															
	W																
0x08C OPP3	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	256KOPP															
	W																
0x090 TMD	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

35.2.3 Register descriptions

This section consists of descriptions of the user mode registers.

35.2.3.1 Module Configuration Register (MCR)

The following field and bit descriptions fully define the MCR register ([Figure 391](#)).

Note: A number of Module Configuration Register (MCR) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in [Section 35.3, "Functional Description"](#).

Figure 391. MCR Register

Offset 0x0000												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RVE	RRE	AEE	EEE	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c	w1c	w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	FERS	0	0	PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	0/1	1	0	0	0	0	0	0	0	0	0

MCR register functions are as shown in [Table 460](#).

Table 460. MCR field descriptions

Field	Description
0 RVE	Read Voltage Error RVE provides information on previous reads monitoring the read pump voltage. If the read voltage is detected to be out of range, this bit will set to signify that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without voltage issues 1 A previous read may have been corrupted due to read voltage being out of range
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the read reference. If the read reference is detected to be out of range, this bit will set to signify that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without reference issues 1 A previous read may have been corrupted due to read reference being out of range

Table 460. MCR field descriptions (continued)

Field	Description
2 AEE	Address Encode Error AEE provides information on previous reads monitoring the address encode feature. On every read request to the flash, the incoming address is compared to an encoded address (row, column, and block) coming back from the memory array using the read data sense amp timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error will be recorded. If an address encode mismatch is detected, this bit will set to signify that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without address encode mismatches 1 A previous read may be corrupted based on address encode mismatch
3 EEE	ECC after ECC Error EEE provides information on previous reads monitoring the ECC after ECC feature. On every read request to the flash, ECC is recalculated serially, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error will be reported. If an ECC after ECC mismatch is detected, this bit will set to signify that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without ECC after ECC mismatches 1 A previous read may be corrupted based on ECC calculation errors
4–15	Reserved, reset to 0.
16 EER	ECC Event Error This bit provides information on previous reads. If a double bit detection occurred, the EER bit is set to a 1. This bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be set by software. In the event of a single bit detection and correction, this bit is not set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally 1 An ECC Error occurred during a previous read
17 RWE	Read-While-Write Event Error This bit provides information on previous read-while-write (RWW) reads. If an RWW error occurs, this bit is set to 1. The bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be written to a 1. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally 1 A read-while-write error occurred during a previous read
18 SBC	Single Bit Correction SBC provides information on previous reads, if the SBCE is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without corrections 1 A Single Bit Correction occurred during a previous read
19	Reserved, reset to 0.

Table 460. MCR field descriptions (continued)

Field	Description
20 PEAS	Program Access Space PEAS indicates which space is valid for program or erase operations — either main array space or UTest NVM space. It should be noted that if the flash is sealed, and the Test Mode Disable Seal is programmed, erase is not allowed on the UTest NVM space. PEAS = 0 indicates the main address space is active for all program or erase operations. PEAS = 1 indicates the UTest NVM address space is active for program or erase (if applicable) operations. The value in PEAS is captured and held when the UTest NVM block is enabled with the first interlock write done for program operations (and erase operations if the flash is unsealed). The value of PEAS is retained between sampling events (in other words, subsequent first interlock writes). The value in PEAS may be changed during erase-suspended program, and reverts back to its original state once the erase-suspended program is completed. PEAS is read only. 0 UTest NVM address space is disabled for program/erase and main address space enabled 1 UTest NVM address space is enabled for program/erase and main address space disabled
21 DONE	State Machine Status DONE indicates whether the embedded flash memory is performing a high voltage operation. DONE is set to a 1 on termination of the embedded flash memory reset. DONE is read only. DONE is set to a 1 at the end of program and erase high voltage sequences. 0 Flash memory is executing a high voltage operation 1 Flash memory is not executing a high voltage operation
22 PEG	Program/Eraser Good The PEG bit indicates the completion status of the last flash memory program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the embedded flash memory is reset. PEG is read only. The value of PEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation. PEG will be 0 if an Erase-Suspended Program is attempted to a block that has been selected for Erase. PEG will also be 0 in the event of an error in the interlock write for program only. An error in the interlock write occurs when the ECC value provided to the flash memory does not match the ECC calculated within the flash memory. In the event of this, high voltage will not be applied to the array, and PEG will indicate a failed program. 0 Program or erase operation failed 1 Program or erase operation successful <i>Note: If program or erase operations are attempted on blocks that are locked, the response from embedded flash memory is PEG = 1, indicating that the operation was successful, and the contents of the block are properly protected from the program or erase operation. PEG = 1 is also true if an abort occurs during an HV request to a locked block.</i> <i>Note: If a program operation is attempted to a location marked as OTP, the response from the embedded flash memory is PEG = 0, indicating that the operation was not allowed, and the value interlocked was not programmed, since the desired doubleword was already programmed with a previous program operation.</i>

Table 460. MCR field descriptions (continued)

Field	Description
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag due to a high voltage event (program or erase) finishing (normal, abort, suspend). If PECIE is written while not in a high voltage event, the interrupt will not immediately trigger, but will trigger after the next high voltage event is completed. 0 An interrupt request will not be generated when the DONE flag is set 1 An interrupt request will be generated when the DONE flag is set
24 FERS	Factory Erase and Program FERS is used for erase and program operations to enable a faster operation. Factory Erase must be done following "Initial Factory Conditions All Temps" with respect to voltage conditions, write/erase cycling, and temperature which can be from –40 °C to 150 °C. After setting the ERS or PGM bit, but prior to setting EHV, FERS may be set to enable a Factory Erase or Program. FERS can be written only when ERS or PGM are high, but EHV is still low. Factory Erase may also be disabled by writing a location in the UTest NVM space. Address 0x0020 in the UTest NVM space will be checked at reset, and if programmed with data that contains at least one zero, a factory erase is not allowed, and FERS is locked. Writes attempted that violate the above cases will result in FERS remaining cleared. FERS is cleared on reset. 0Factor Erase and Program is disabled 1Factory Erase and Program is enabled
25–26	Reserved, reset to 0.
27 PGM	Program PGM is used to set up embedded flash memory for a program operation. A 0 to 1 transition of PGM initiates a program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions: – User mode read (ERS is low and UTE is low) – Erase suspend (ERS and ESUS are 1) with EHV low PGM can be cleared only when PSUS and EHV are low and DONE is high. PGM is cleared on reset. 0Flash memory is not executing a program sequence 1Flash memory is executing a program sequence
28 PSUS	Program Suspend PSUS is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state. The embedded flash memory is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the embedded flash memory in program suspend. PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to program. The embedded flash memory cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset. 0Program sequence is not suspended 1Program sequence is suspended

Table 460. MCR field descriptions (continued)

Field	Description
29 ERS	Erase ERS is used to set up embedded flash memory for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set in user mode read (PGM is low and UTE is low). ERS can be cleared only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0Flash memory is not executing an erase sequence 1Flash memory is executing an erase sequence
30 ESUS	Erase Suspend ESUS is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state. The embedded flash memory is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in erase suspend. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to erase. The embedded flash memory cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0Erase sequence is not suspended 1Erase sequence is suspended
31 EHV	Enable High Voltage The EHV bit enables the embedded flash memory for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock under one of the following conditions: – Erase (ERS = 1, ESUS = 0) – Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0) – Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0) If a program operation is to be initiated while an erase is suspended, EHV must be cleared while in erase suspend before setting PGM. In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS low, and ESUS low terminates the current program/erase high-voltage operation. In an erase-suspended program operation, a 1 to 0 transition of EHV with DONE high, ESUS high, PGM high, ERS high, and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. EHV may not be written to a 1 after an abort is requested (EHV being cleared) and before DONE transitions high. A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the embedded flash memory to exit suspend. EHV may not be written after a suspend bit is set high and before DONE transitions high. EHV may not be set low after the current suspend bit is set low and before DONE transitions low. 0Flash memory is not enabled to perform a high voltage operation 1Flash memory is enabled to perform a high voltage operation <i>Note: Aborting a high voltage operation leaves addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.</i>

Note: See [Section 35.3, “Functional Description](#) for information about simultaneous MCR register writes, and priority levels.

35.2.3.2 Alternate Module Configuration Register (MCRA)

The following field and bit descriptions fully define the MCRA register ([Figure 392](#)).

Note: A number of Alternate Module Configuration Register (MCRA) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in [Section 35.3](#), “Functional Description”.

Figure 392. MCRA Register

Offset 0x0004												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PEASa	DONEa	PEGa	0	0	0	0	PGMa	PSUSa	ERSa	ESUSa	EHVa
W																
Reset	0	0	0	0	0	0/1	1	0	0	0	0	0	0	0	0	0

MCRA register functions are shown in [Table 461](#).

Table 461. MCRA field descriptions

Field	Description
	Reserved, reset to 0.
PEASa	Program Access Space PEASa is used to indicate which space is valid for program operations, either main array space or UTest NVM space for the Alternate program and erase interface. It should be noted that erase is not allowed to the UTest NVM space (even if unsealed with the Test Mode Disable Seal) through the alternate interface. If an interlock write is done to the UTest NVM space with ERSa set, PEASa will remain 0. See PEAS definition in the MCR register for more information (Table 460).
DONEa	State Machine Status DONEa indicates if the embedded flash memory is performing a high-voltage operation for the Alternate program and erase interface. See DONE definition in the MCR register for more information (Table 460).
PEGa	Program/Erase Good The PEGa bit indicates the completion status of the last flash memory program or erase sequence for which high-voltage operations were initiated for the Alternate program and erase interface. See PEG definition in the MCR register for more information (Table 460). It should be noted that in the event of a error on the alternate interface PEGa will clear, but the ADR register will not be updated with the failing location. The ADR register is only valid for main interface program or erase fails, and other fails noted in the ADR register.

Table 461. MCRA field descriptions (continued)

Field	Description
	Reserved, reset to 0.
PGMa	Program PGMa is used to set up embedded flash memory for a program operation for the Alternate program and erase interface. See PGM definition in the MCR register for more information (Table 460).
PSUSa	Program Suspend PSUSa is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state for the Alternate program and erase interface. See PSUS definition in the MCR register for more information (Table 460).
ERSa	Erase ERSa is used to set up embedded flash memory for an erase operation for the Alternate program and erase interface. See ERS definition in the MCR register for more information (Table 460).
ESUSa	Erase Suspend ESUSa is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state for the Alternate program and erase interface. See ESUS definition in the MCR register for more information (Table 460).
EHVa	Enable High Voltage The EHVa bit enables the embedded flash memory for a high-voltage program/erase operation for the Alternate program and erase interface. See EHV definition in the MCR register for more information (Table 460).

Note: See [Section 35.3, “Functional Description](#) for information about simultaneous MCRA register writes, and priority levels.

35.2.3.3 Extended Module Configuration Register (MCRE)

The following field and bit descriptions fully define the MCRE register ([Figure 393](#)).

Figure 393. MCRE Register

Offset 0x0008																Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	HT	0	0	n256K				n64Kh				n32Kh		n16Kh																	
W																															
Reset ⁽¹⁾	0	0	0	—	—	—	—	—	—	—	—	—	—	—	—	—															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	n64Km				n32Km				n16Km				n64Kl				n32Kl		n16Kl												
W																															
Reset ⁽¹⁾	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—															

1. A reset value of “—” indicates that the value is set at the factory and can vary among different devices.

MCRE register functions are as shown in [Table 462](#).

Table 462. MCRE field descriptions

Field	Description
0 HT	High Temperature Enabled HT provides configuration information of the module. If the flash is tuned for high temperature operation (165 °C) at reduced performance (slower read at 165 °C or less and slower program and erase at 165 °C or less), and limited features at this temperature (no factory erase or margin reads), the HT status bit will be a 1. Default is 150 °C as maximum operating temperature, and the HT status bit will be a 0. HT is read only. 0 150 °C maximum operating temperature 1 165 °C maximum operating temperature with reduced performance
1–2	Reserved, reset to 0.
3–7 n256K[4:0]	Number of 256 KB blocks. The value of the n256K field depends on the size of the embedded flash memory. n256K is read only. 00000 Zero 256 KB blocks 00001 Two 256 KB blocks (512 KB total) 00010 Four 256 KB blocks (1 MB total) 00011 Six 256 KB blocks (1.5 MB total) 00100 Eight 256 KB blocks (2 MB total) 00101 Ten 256 KB blocks (2.5 MB total) 00110 Twelve 256 KB blocks (3 MB total) 00111 Fourteen 256 KB blocks (3.5 MB total) 01000 Sixteen 256 KB blocks (4 MB total) 01001 Eighteen 256 KB blocks (4.5 MB total) 01010 Twenty 256 KB blocks (5 MB total) 01011 Twenty-two 256 KB blocks (5.5 MB total) 01100 Twenty-four 256 KB blocks (6 MB total) 01101 Twenty-six 256 KB blocks (6.5 MB total) 01110 Twenty-eight 256 KB blocks (7 MB total) 01111 Thirty 256 KB blocks (7.5 MB total) 10000 Thirty-two 256 KB blocks (8 MB total) 10001 Reserved 10010 Reserved 10011 Reserved 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

Table 462. MCRE field descriptions (continued)

Field	Description
8–10 n64Kh[2:0]	Number of 64 KB blocks in partitions 4 and 5. n64Kh is read only. 000 Zero 64 KB blocks 001 Two 64 KB blocks 010 Four 64 KB blocks 011 Six 64 KB blocks 100 Eight 64 KB blocks 101 Twelve 64 KB blocks (see note) 110 Sixteen 64 KB blocks (see note) 111 Reserved <i>Note: If Twelve or Sixteen 64 KB blocks are selected in the high partition pair, then no other 16 KB or 32 KB blocks are allowed in the high partition pair.</i>
11–12 n32Kh[1:0]	Number of 32 KB blocks in partitions 4 and 5. n32K is read only. 00 Zero 32 KB blocks 01 Two 32 KB blocks 10 Four 32 KB blocks 11 Reserved
13–15 n16Kh[2:0]	Number of 16 KB blocks in partition 4 and 5. n16K is read only. 000 Zero 16 KB blocks 001 Two 16 KB blocks 010 Four 16 KB blocks 011 Six 16 KB blocks 100 Eight 16 KB blocks 101 Reserved 110 Reserved 111 Reserved
16–18 n64Km[2:0]	Number of 64 KB blocks in partitions 2 and 3. n64Kh is read only. 000 Zero 64 KB blocks 001 Two 64 KB blocks 010 Four 64 KB blocks 011 Six 64 KB blocks 100 Eight 64 KB blocks 101 Reserved 110 Reserved 111 Reserved
19–20 n32Km[1:0]	Number of 32 KB blocks in partitions 2 and 3. n32K is read only. 00 Zero 32 KB blocks 01 Two 32 KB blocks 10 Four 32 KB blocks 11 Reserved
21–23 n16Km[2:0]	Number of 16 KB blocks in partition 2 and 3. n16K is read only. 000 Zero 16 KB blocks 001 Two 16 KB blocks 010 Four 16 KB blocks 011 Six 16 KB blocks 100 Eight 16 KB blocks 101 Reserved 110 Reserved 111 Reserved

Table 462. MCRE field descriptions (continued)

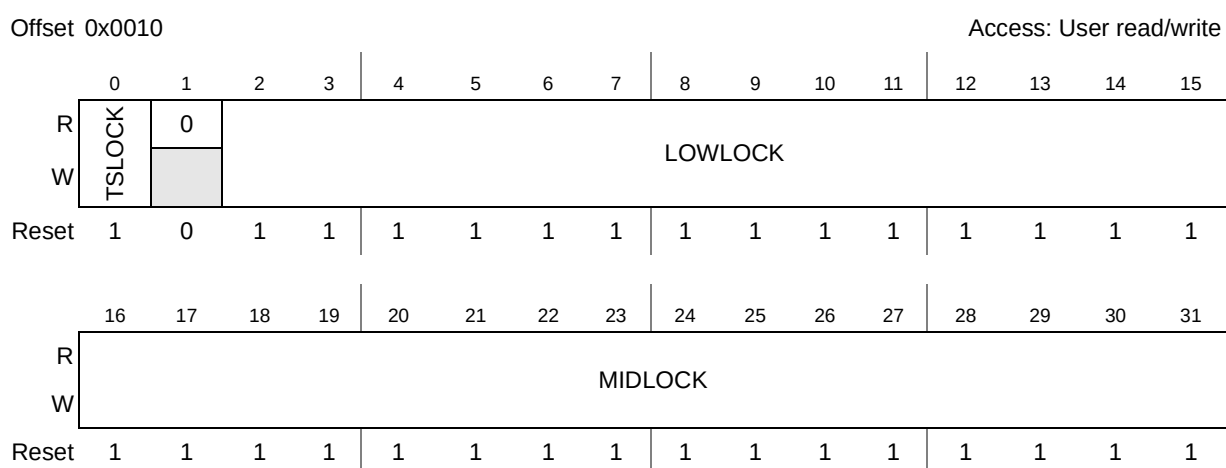
Field	Description
24–26 n64Kl[2:0]	Number of 64 KB blocks in partitions 0 and 1. n64Kh is read only. 000 Zero 64 KB blocks 001 Two 64 KB blocks 010 Four 64 KB blocks 011 Six 64 KB blocks 100 Eight 64 KB blocks 101 Reserved 110 Reserved 111 Reserved
27–28 n32Kl[1:0]	Number of 32 KB blocks in partitions 0 and 1. n32K is read only. 00 Zero 32 KB blocks 01 Two 32 KB blocks 10 Four 32 KB blocks 11 Reserved
29–31 n16Kl[2:0]	Number of 16 KB blocks in partition 0 and 1. n16K is read only. 000 Zero 16 KB blocks 001 Two 16 KB blocks 010 Four 16 KB blocks 011 Six 16 KB blocks 100 Reserved 101 Reserved 110 Reserved 111 Reserved

35.2.3.4 Lock 0 register (LOCK0)

The Lock 0 (LOCK0) register provides a means to protect blocks from being modified.

The following field and bit descriptions fully define the LOCK0 register ([Figure 394](#)).

Figure 394. LOCK0 register



LOCK0 register functions are as shown in [Table 463](#).

Table 463. LOCK0 field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCK register signifies that the UTest NVM block is locked for program. A value of 0 in the TSLOCK register signifies that the UTest NVM block is available to receive program pulses. The TSLOCK register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, TSLOCK register is not writable if a high voltage operation is suspended. 0 UTest NVM block is available to be programmed 1 UTest NVM block is locked and cannot be programmed
1	Reserved, reset to 0.

Table 463. LOCK0 field descriptions (continued)

Field	Description
2–15 LOWLOCK[13:0]	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. The default value of the LOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations 1 Block is locked and unavailable for program and erase operations

Table 463. LOCK0 field descriptions (continued)

Field	Description
16–31 MIDLOCK[15:0]	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations 1 Block is locked and unavailable for program and erase operations

35.2.3.5 Lock 1 Register (LOCK1)

The Lock 1 (LOCK1) register provides a means to protect blocks from being modified.

Figure 395. LOCK1 Register

Offset 0x0014												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHLOCK															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LOCK1 register functions are as shown in [Table 464](#).

Table 464. LOCK1 field descriptions

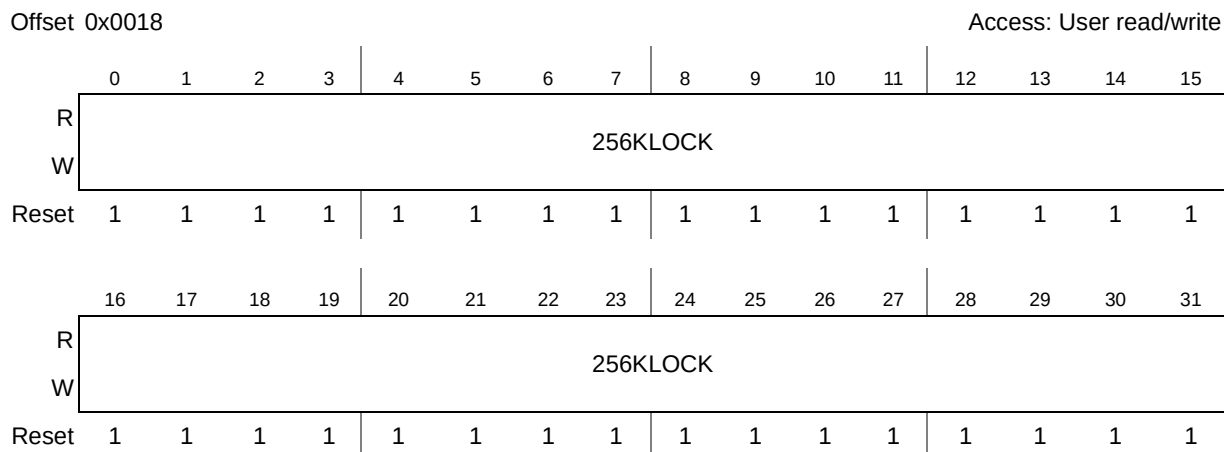
Field	Description
0–15	Reserved, reset to 0.
16–31 HIGHLOCK[15:0]	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations 1 Block is locked and unavailable for program and erase operations

35.2.3.6 Lock 2 Register (LOCK2)

The Lock 2 (LOCK2) register provides a means to protect blocks from being modified.

The following field and bit descriptions fully define the LOCK2 register ([Figure 396](#)).

Figure 396. LOCK2 Register



LOCK2 register functions are as shown in [Table 465](#).

Table 465. LOCK2 field descriptions

Field	Description
0–31 256KLOCK[31:0]	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations 1 Block is locked and unavailable for program and erase operations

35.2.3.7 Lock 3 Register (LOCK3)

The Lock 3 (LOCK3) register provides a means to protect blocks from being modified.

The following field and bit descriptions fully define the LOCK3 register ([Figure 397](#)).

Figure 397. LOCK3 Register

Offset 0x001C												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256KLOCK															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LOCK3 register functions are as shown in [Table 466](#).

Table 466. LOCK3 field descriptions

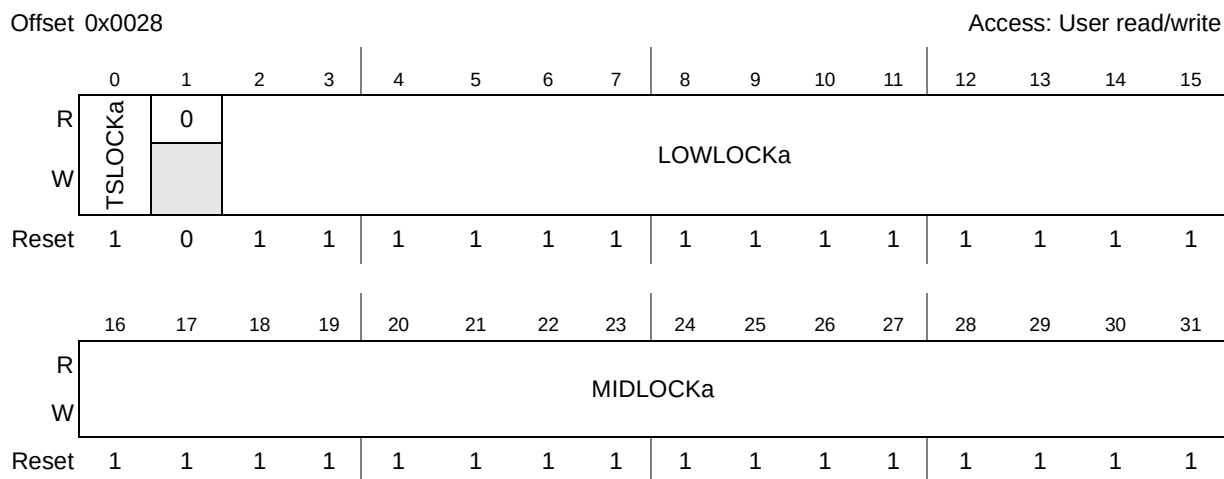
Field	Description
0–15	Reserved, reset to 0.
16–31 256KLOCK[47:32]	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations 1 Block is locked and unavailable for program and erase operations

35.2.3.8 Alternate Lock 0 Register (LOCK0a)

The Alternate Lock 0 (LOCK0a) register provides a means to protect blocks from being modified on the alternate program and erase interface.

The following field and bit descriptions fully define the LOCK0a register ([Figure 398](#)).

Figure 398. LOCK0a Register



LOCK0a register functions are as shown in [Table 467](#).

Table 467. LOCK0a field descriptions

Field	Description
TSLOCKa	Alternate UTest NVM Lock This bit is used to lock the UTest NVM block from programs on the alternate program and erase interface (erase protection is not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCKa register signifies that the UTest NVM block is locked for program on the alternate interface. A value of 0 in the TSLOCKa register signifies that the UTest NVM block is available to receive program pulses on the alternate interface. The TSLOCKa register is not writable once an interlock write is completed until DONEa is set at the completion of the requested operation. Likewise, TSLOCKa register is not writable if a high-voltage operation is suspended. The TSLOCKa bit may be written as a register. The default value of the TSLOCKa bits is locked. 0 The UTest NVM block is available to receive program pulses on the alternate interface 1 The UTest NVM block is locked for program on the alternate interface
	Reserved, reset to 0

Table 467. LOCK0a field descriptions (continued)

Field	Description
LOWLOCKa[13:0]	Alternate Low Block Lock This is the method to lock blocks for the alternate program and erase interface. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCKa[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONEa is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. The default value of the LOCKa bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCKa bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations 1 Block is locked and unavailable for program and erase operations

Table 467. LOCK0a field descriptions (continued)

Field	Description
MIDLOCKa[15:0]	Alternate Mid Block Lock This is the method to lock blocks for the alternate program and erase interface. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCKa[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The lock register is not writable once an interlock write is completed until DONEa is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCKa bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCKa bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations 1 Block is locked and unavailable for program and erase operations

35.2.3.9 Alternate Lock 1 Register (LOCK1a)

The Alternate Lock 1 (LOCK1a) register provides a means to protect blocks from being modified on the alternate program and erase interface.

The following field and bit descriptions fully define the LOCK1a register ([Figure 399](#)).

Figure 399. LOCK1a Register

Offset 0x002C																Access: User read/write													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15													
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31													
R	HIGHLOCKa																												
W																													
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1													

LOCK1a register functions are as shown in [Table 468](#).

Table 468. LOCK1a field descriptions

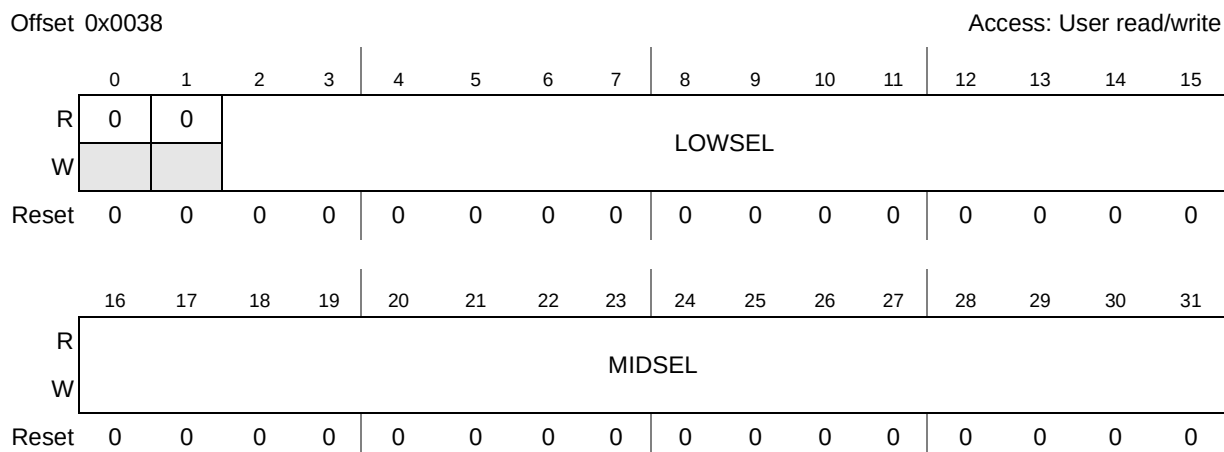
Field	Description
	Reserved, reset to 0.
HIGHLOCKa[15:0]	Alternate High Block Lock This is the method to lock blocks for the alternate program and erase interface. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCKa[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONEa is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCKa bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCKa bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations 1 Block is locked and unavailable for program and erase operations

35.2.3.10 Select 0 Register (SEL0)

The Select 0 (SEL0) register provides a means to select blocks to be operated on during erase.

The following field and bit descriptions fully define the SEL0 register ([Figure 400](#)).

Figure 400. SEL0 Register



SEL0 register functions are as shown in [Table 469](#).

Table 469. SEL0 field descriptions

Field	Description
0–1	Reserved, reset to 0.
2–15 LOWSEL[13:0]	LOW Block Select A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Low Blocks starts at LOWSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. LOWSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0The block is not selected 1The block is selected for erase

Table 469. SEL0 field descriptions (continued)

Field	Description
16–31 MIDSEL[15:0]	Mid Block Select A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Mid Blocks starts at MIDLSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. Mid Blocks exist in partitions 2 and 3. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. MIDSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0The block is not selected 1The block is selected for erase

35.2.3.11 Select 1 Register (SEL1)

The Select 1 (SEL1) register provides a means to select blocks to be operated on during erase.

The following field and bit descriptions fully define the SEL1 register ([Figure 401](#)).

Figure 401. SEL1 Register

Offset 0x003C												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEL1 register functions are as shown in [Table 470](#).

Table 470. SEL1 field descriptions

Field	Description
0–15	Reserved, reset to 0.
16–31 HIGHSEL[15:0]	High Block Select A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the High Blocks starts at HIGHSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. HIGHSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0The block is not selected 1The block is selected for erase

35.2.3.12 Select 2 Register (SEL2)

The Select 2 (SEL2) register provides a means to select blocks to be operated on during erase.

The following field and bit descriptions fully define the SEL2 register ([Figure 402](#)).

Figure 402. SEL2 Register

Offset 0x0040												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	256KSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256KSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEL2 register functions are as shown in [Table 471](#).

Table 471. SEL2 field descriptions

Field	Description
0–31 256KSEL[31:0]	256 KB Block Select A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0The block is not selected 1The block is selected for erase

35.2.3.13 Select 3 Register (SEL3)

The Select 3 (SEL3) register provides a means to select blocks to be operated on during erase.

The following field and bit descriptions fully define the SEL3 register ([Figure 403](#)).

Figure 403. SEL3 Register

Offset 0x0044												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256KSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEL3 register functions are as shown in [Table 472](#).

Table 472. SEL3 field descriptions

Field	Description
0–15	Reserved, reset to 0.
16–31 256KSEL[47:32]	256 KB Block Select A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

35.2.3.14 Address Register (ADR)

The Address (ADR) register provides the first failing address in the event of a failure (ECC or PGM/Erase state machine), as well as single bit correction information.

The following field and bit descriptions fully define the ADR register ([Figure 404](#)).

Figure 404. ADR Register

Offset 0x0050																Access: User read/write													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15													
R	SAD	aH	aM	aL	a256	a64k	a32k	a16k	ADDR																				
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31													
R	ADDR													0	0	0													
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													

ADR register functions are as shown in [Table 473](#).

Table 473. ADR field descriptions

Field	Description
0 SAD	UTest NVM Address The SAD bit qualifies the address captured during an ECC Event Error, Single Bit Correction, or State Machine operation. SAD has the same characteristics as ADDR related to capturing of addresses. The SAD register is not writable. 0 Address captured is from main array space 1 Address captured is from UTest NVM array space
1 aH	Address High region The aH bit qualifies the address field (ADDR) to the region of a16K, a32K, and a64k. If aH is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from high address region 1 Address captured or to be accessed is from high address region
2 aM	Address Mid region The aM bit qualifies the address field (ADDR) to the region of a16K, a32K, and a64k. If aM is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from mid address region 1 Address captured or to be accessed is from mid address region
3 aL	Address Low region The aL bit qualifies the address field (ADDR) to the region of a16K, a32K, and a64k. If aL is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from low address region 1 Address captured or to be accessed is from low address region
4 a256k	Address 256 KB region The a256k bit qualifies the address field (ADDR) to the region. If a256k is set, then the ADDR field maps to that region. See ADDR description for more information. The a256k register is writable. 0 Address captured or to be accessed is not from 256 KB region 1 Address captured or to be accessed is from 256 KB region
4 a64k	Address 64 KB region The a64k bit qualifies the address field (ADDR) to the region. If a64k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 64 KB region 1 Address captured or to be accessed is from 64 KB region
6 a32k	Address 32 KB region The a32k bit qualifies the address field (ADDR) to the region. If a32k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 32 KB region 1 Address captured or to be accessed is from 32 KB region
7 a16k	Address 16 KB region The a16k bit qualifies the address field (ADDR) to the region. If a16k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 16 KB region 1 Address captured or to be accessed is from 16 KB region

Table 473. ADR field descriptions

Field	Description
8–28 ADDR[23:3]	Address The ADR register provides the first failing address in the event of ECC event error (EER set), Single Bit Correction (SBC set), as well as providing the address of a failure that may have occurred in a state machine operation (PEG cleared). ECC event errors take priority over Single Bit Corrections, which take priority over state machine errors. This is especially valuable in the event of an RWW operation, where the read senses an ECC error or Single Bit correction, and the state machine fails simultaneously. The failing address for ECC event error is held until EER is cleared. The failing address for Single Bit Correction is held until an ECC event error, or until SBC is cleared. State machine address is held until an ECC event error or Single Bit Correction event occurs, or until the next state machine event (PEG cleared) occurs. This address is always a doubleword address that selects 64 bits. ADDR[23:3] is an offset from a base address of 0x0 for each block size region. The aH, aM, aL, a16k, a32k, a64k, and a256k qualify the block size region the ADDR field.
29–31	Reserved, reset to 0.

35.2.3.15 User Test 0 Register (UT0)

The User Test 0 (UT0) register provides a means to control UTest. UTest mode gives the ability to perform test features on the flash memory. This register is only writable when the flash memory is put into UTest mode by writing a passcode.

The following field and bit descriptions fully define the UT0 register ([Figure 405](#)).

Figure 405. UT0 Register

Offset 0x0054												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UTE	SBCE	0	0	0	0	0	0	0	0	0	0	0	CPR	CPA	CPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	NAIBP	AIBPE	0	AISUS	MRE	MRV	0	AIS	AIE	AID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

UT0 register functions are as shown in [Table 474](#).

Table 474. UT0 field descriptions

Field	Description
0 UTE	U-Test Enable This status bit indicates when U-Test is enabled. All bits in UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8, and UM9 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect a status of enabled, and is enabled until it is cleared by a register write. The UTE password will only be accepted if PGM = 0 and ERS = 0 (program and erase are not being requested). UTE can only be cleared if AID = 1, MRE and AIE = 0. While clearing UTE, writes to set AIE or MRE will be ignored. For UTE, the password 0xF9F9_9999 must be written to the UT0 register.
1 SBCE	Single Bit Correction Enable SBCE enables Single Bit Correction results to be observed in SBC. ECC corrections that occur when SBCE is cleared will not be logged. 0 Single Bit Correction observation is disabled 1 Single Bit Correction observation is enabled
2–12	Reserved, reset to 0
13 CPR	Customer Programmable Read Voltage and Reference Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Read Voltage and Reference Error detection will be noted. This bit may not be set while CPE and CPA are set. If CPE or CPA are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled
14 CPA	Customer Programmable Address Encode Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Address Encode Error detection will be noted. This bit may not be set while CPE and CPR are set. If CPE or CPR are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a EDC after ECC Error detection will be noted. This bit may not be set while CPR and CPA are set. If CPR or CPA are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
16–21	Reserved, reset to 0

Table 474. UT0 field descriptions (continued)

Field	Description
22 NAIBP	Next Array Integrity Break Point If AIBPE is set, NAIBP will be set when a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to 1, but may be cleared to 0. Clearing NAIBP to 0 will enable the Array Integrity operation to be restarted after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID will be set with NAIBP remaining 0. 0 Array Integrity state machine is not currently at a break point 1 Array Integrity state machine is at a break point
23 AIBPE	Array Integrity Break Point Enable If you want to enable breakpoints during an array integrity test, AIBPE may be set. See NAIBP description for more information about array integrity breakpoints. 0 Array Integrity breakpoints are not enabled 1 Array Integrity breakpoints are enabled during Array Integrity checks
24	Reserved, reset to 0
25 AISUS	Array Integrity Suspend AISUS enables a suspend of an array integrity operation. Array integrity may be suspended by setting AISUS, and resumed by clearing AISUS. AISUS can be written to a 1 only when AID is low. AISUS can be written to a 0 only when AID is high. Attempting to write AISUS and AIE on the same clock cycle will result in only AIE getting written. 0 Array integrity sequence not suspended 1 Array integrity sequence is suspended
26 MRE	Margin Read Enable MRE combined with MRV enables user margin reads to be done. Normal user reads are not affected by MRE, although user reads while the margin read operation is ongoing are not supported. MRE is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 0 Margin reads are not enabled 1 Margin reads are enabled
27 MRV	Margin Read Value MRV selects the margin level that is being checked. The margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 0 Zero's margin reads are requested 1 One's margin reads are requested
28	Reserved, reset to 0
29 AIS	Array Integrity Sequence AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences that normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is logically sequential. It should be noted that the time to run a sequential sequence is shorter than the time to run the proprietary sequence. AIS is not writeable if AIE is high. 0 Array integrity sequence is proprietary sequence 1 Array integrity sequence is sequential

Table 474. UT0 field descriptions (continued)

Field	Description
30 AIE	Array Integrity Enable AIE set to one starts the array integrity check done on all selected blocks. The address sequence is determined by AIS, and the MISR (UM0 through UM9) can be checked after the operation is complete, to determine if a correct signature has been obtained. Once an array integrity operation is requested (AIE=1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE remains at 1 when the Array Integrity Check, or Margin Read are suspended or are in a breakpoint, and cannot be cleared. AIE is not simultaneously writable to a 1 as UTE is being cleared to a 0. 0 Array integrity checks are not enabled 1 Array integrity checks are enabled
31 AID	Array Integrity Done AID is cleared upon an array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set), an abort is requested or a suspend is requested. AID cannot be written, and is status only. 0 Array integrity check is ongoing 1 Array integrity check is done

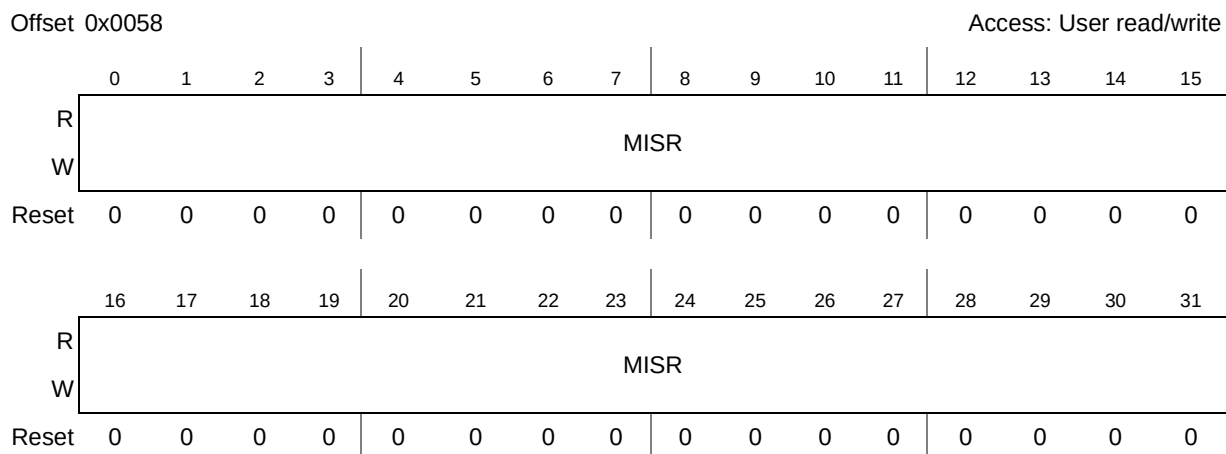
35.2.3.16 User Multiple Input Signature Register

The Multiple Input Signature registers (UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8, and UM9) provide a means to evaluate array integrity.

35.2.3.16.1 UM0 Register

The following field and bit descriptions fully define the UM0 register ([Figure 406](#)).

Figure 406. UM0 Register



MISR register functions are as shown in [Table 475](#).

Table 475. UM0 field descriptions

Field	Description
0–31 MISR[31:0]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]), is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.2 UM1 Register

The following field and bit descriptions fully define the UM1 register ([Figure 407](#)).

Figure 407. UM1 Register

Offset 0x005C												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 476](#).

Table 476. UM1 field descriptions

Field	Description
0–31 MISR[63:32]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]) is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.3 UM2 Register

The following field and bit descriptions fully define the UM2 register ([Figure 408](#)).

Figure 408. UM2 Register

Offset 0x0060												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 477](#).

Table 477. UM2 Field Descriptions

Field	Description
0–31 MISR[95:64]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]), is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.4 UM3 Register

The following field and bit descriptions fully define the UM3 register ([Figure 409](#)).

Figure 409. UM3 Register

Offset 0x0064												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 478](#).

Table 478. UM3 field descriptions

Field	Description
0–31 MISR[127:96]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]), is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.5 UM4 Register

The following field and bit descriptions fully define the UM4 register ([Figure 410](#)).

Figure 410. UM4 Register

Offset 0x0068												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 479](#).

Table 479. UM4 field descriptions

Field	Description
0–31 MISR[159:128]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]) is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.6 UM5 Register

The following field and bit descriptions fully define the UM5 register ([Figure 411](#)).

Figure 411. UM5 Register

Offset 0x006C												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 480](#).

Table 480. UM5 field descriptions

Field	Description
0–31 MISR[191:160]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]) is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.7 UM6 Register

The following field and bit descriptions fully define the UM6 register ([Figure 412](#)).

Figure 412. UM6 Register

Offset 0x0070												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 481](#).

Table 481. UM6 field descriptions

Field	Description
0–31 MISR[223:192]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]) is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.8 UM7 Register

The following field and bit descriptions fully define the UM7 register ([Figure 413](#)).

Figure 413. UM7 Register

Offset 0x0074												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 482](#).

Table 482. UM7 field descriptions

Field	Description
0–31 MISR[255:224]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]) is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.9 UM8 Register

The following field and bit descriptions fully define the UM8 register ([Figure 414](#)).

Figure 414. UM8 Register

Offset 0x0078												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 483](#).

Table 483. UM8 field descriptions

Field	Description
0–31 MISR[287:256]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]) is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.16.10 UM9 Register

The following field and bit descriptions fully define the UM9 register ([Figure 415](#)).

Figure 415. UM9 Register

Offset 0x007C												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MISR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISR register functions are as shown in [Table 484](#).

Table 484. UM9 field descriptions

Field	Description
0–30	Reserved, reset to 0.
31 MISR[288]	The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. <i>Note: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way.</i> The MISR register provides a means to calculate an MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then exclusive ORing the new data. In addition the most significant bit (in this case MISR[288]) is then exclusive ORed into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2], and MISR[0]. The result of the exclusive OR is shifted left on each read. The MISR register is used in array integrity operations. If, during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size), then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

35.2.3.17 Over-Program Protection 0 Register (OPP0)

The Over-Program Protection 0 (OPP0) register provides a means to protect blocks from being over-programmed. This register shows the status of the over-program protection status. For more information on Over-Program Protection see [Section 35.3.1.2.4, Over-program protection enable](#).

The following field and bit descriptions fully define the OPP0 register ([Figure 416](#)).

Figure 416. OPP0 Register

Offset 0x0080																Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	0	0	LOWOPP																												
W																															
Reset ⁽¹⁾	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	MIDOPP																														
W																															
Reset ⁽¹⁾	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—															

1. A reset value of “–” indicates that the value is set at the factory and can vary among different devices.

OPP0 register functions are as shown in [Table 485](#).

Table 485. OPP0 field descriptions

Field	Description
0–1	Reserved, reset to 0.
2–15 LOWOPP[13:0]	Low Block Over-Program Protection A value of 1 in a bit of the Over-Program Protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The LOWOPP register is not writable, and is status only. The default value of the LOWOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming 0 The corresponding block is available to be over-programmed

Table 485. OPP0 field descriptions (continued)

Field	Description
16–31 MIDOPP[15:0]	Mid Block Over-Program Protection A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The MIDOPP register is not writable, and is status only. The default value of the MIDOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming 0 The corresponding block is available to be over-programmed

35.2.3.18 Over-Program Protection 1 Register (OPP1)

The Over-Program Protection 1 (OPP1) register provides a means to protect blocks from being over-programmed. This register shows the status of the over-program protection status. For more information on Over-Program Protection see [Section 35.3.1.2.4, Over-program protection enable](#).

The following field and bit descriptions fully define the OPP1 register ([Figure 417](#)).

Figure 417. OPP1 Register

Offset 0x0084								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset ⁽¹⁾	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHOPP															
W																
Reset ⁽¹⁾	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of “–” indicates that the value is set at the factory and can vary among different devices.

OPP1 register functions are as shown in [Table 486](#).

Table 486. OPP1 field descriptions

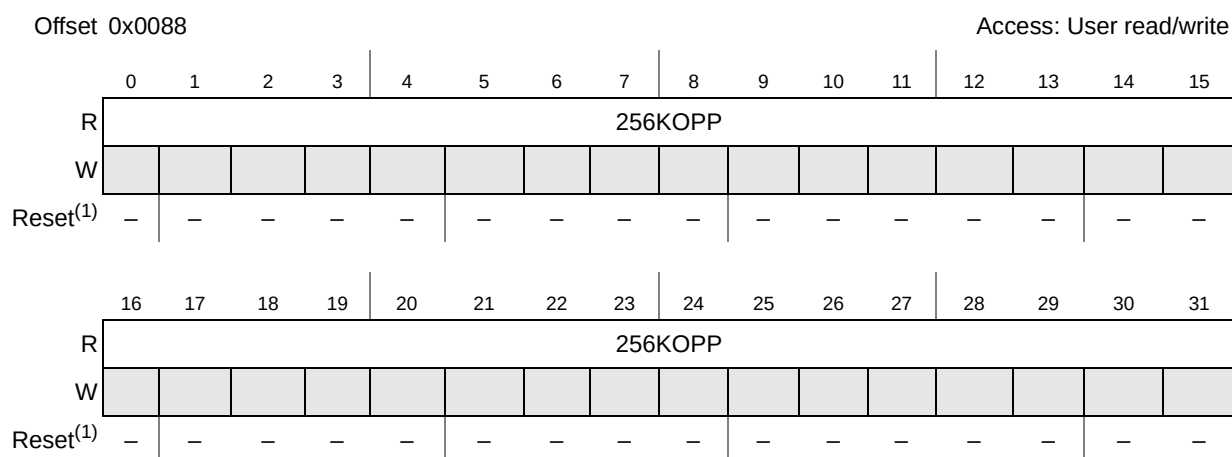
Field	Description
0–15	Reserved, reset to 0.
16–31 HIGHOPP[15:0]	High Block Over-Program Protection A value of 1 in a bit of the Over-Program Protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the High Blocks starts at HIGHOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The HIGHOPP register is not writable, and is status only. The default value of the HIGHOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming 0 The corresponding block is available to be over-programmed

35.2.3.19 Over-Program Protection 2 Register (OPP2)

The Over-Program Protection 2 (OPP2) register provides a means to protect blocks from being over-programmed. This register shows the status of the over-program protection status. For more information on Over-Program Protection see [Section 35.3.1.2.4, Over-program protection enable](#).

The following field and bit descriptions fully define the OPP2 register ([Figure 418](#)).

Figure 418. OPP2 Register



1. A reset value of “–” indicates that the value is set at the factory and can vary among different devices.

OPP2 register functions are as shown in [Table 487](#).

Table 487. OPP2 field descriptions

Field	Description
0–31 256KOPP[31:0]	256K Block Over-Program Protection A value of 1 in a bit of the Over-Program Protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at 256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the 256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming 0 The corresponding block is available to be over-programmed

35.2.3.20 Over-Program Protection 3 Register (OPP3)

The Over-Program Protection 3 (OPP3) register provides a means to protect blocks from being over-programmed. This register shows the status of the over-program protection status. For more information on Over-Program Protection see [Section 35.3.1.2.4, Over-program protection enable](#).

The following field and bit descriptions fully define the OPP3 register ([Figure 419](#)).

Figure 419. OPP3 Register

Offset 0x008C												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset ⁽¹⁾	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256KOPP															
W																
Reset ⁽¹⁾	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of “–” indicates that the value is set at the factory and can vary among different devices.

OPP3 register functions are as shown in [Table 488](#).

Table 488. OPP3 field descriptions

Field	Description
0–15	Reserved, reset to 0.
16–31 256KOPP[47:32]	256K Block Over-Program Protection A value of 1 in a bit of the Over-Program Protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at 256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the 256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming 0 The corresponding block is available to be over-programmed

35.2.3.21 Test Mode Disable Password Check Register (TMD)

The Test Mode Disable Password Check (TMD) register provides a means to provide a challenge password to disable the Test Mode Disable Seal block select. See [Section 35.2.1.2, Test Mode Disable Seal](#) for more information.

The following field and bit descriptions fully define the TMD register ([Figure 420](#)).

Figure 420. TMD Register

Offset 0x0090												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Test Mode Disable Check register functions are as shown in [Table 489](#).

Table 489. TMD field descriptions

Field	Description
0–31 0	The Test Mode Disable Check register is used as a location to provide a password challenge to unlock blocks selected to be sealed as part of the Test Mode Disable seal mechanism. Please see Section 35.2.1.2, Test Mode Disable Seal for more information. Writes to the register have no effect except for a password challenge. Reads to this register will always return 0. The register's reset value is 0.

35.3 Functional Description

The embedded flash memory consists of address spaces in four groupings:

- Low address space
- Mid address space
- High address space
- 256 KB address space

The main address space is divided into partitions. Each address space mentioned above consists of a partition pair. Partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a write (program or erase) to a given partition, it can simultaneously perform a read from any other partition. For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written). For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The main address space is also divided into blocks to implement independent erase and program protection. The UTest NVM block also exists as a block, and has independent program protection. The UTest NVM block is included to support systems that require nonvolatile memory (NVM) for security or to store system initialization information.

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit-by-bit basis in the section [Section 35.2.3.1, "Module Configuration Register \(MCR\)"](#). The write locks do not consider the effects of trying to write two or more bits simultaneously. The embedded flash memory does not allow bits to be written simultaneously which put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in [Table 490](#).

Table 490. MCR bit set/clear priority levels

Priority level	MCR Bit(s)
1	ERS
2	PGM

Table 490. MCR bit set/clear priority levels

Priority level	MCR Bit(s)
3	EHV
4	ESUS, PSUS

If two or more MCR bits are written simultaneously only the bit with the lowest priority number is accepted for modification. Setting two bits with the same priority number is prevented by existing write locks or do not put the flash memory in an illegal state.

For example, setting ERS and PGM simultaneously results in only ERS being set. Attempting to clear EHV while setting PSUS results in EHV being cleared, while PSUS is unaffected.

Each read of the embedded flash memory retrieves a page, or eight consecutive words (256 bits), of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits [4:2]. The flash memory page read architecture easily supports both cache and burst mode at the BIU level for high-speed read applications.

The embedded flash memory supports fault tolerance through error correction code (ECC) and error detection. ECC implemented within the embedded flash memory corrects single-bit failures and detects double-bit failures.

A software mechanism is provided to independently lock and unlock each block.

Program and erase of the embedded flash memory requires multiple system clock cycles to complete. The program and erase sequence may be suspended or aborted.

The embedded flash memory may reside in various modes. The modes that are available include:

- User mode
- Low Power mode
- UTest mode

Each of these modes is discussed in detail in the following sections.

35.3.1 User mode

In user mode the embedded flash memory may be read and written (register writes and interlock writes), programmed, or erased. The following subsections define all actions that may be performed in user mode.

35.3.1.1 Read and write

The default state of the embedded flash memory is read. The main and UTest NVM address space can be read only in the read state. The MCR is always available for read, except when the embedded flash memory is in low power mode. The embedded flash memory enters the read state on reset. The embedded flash memory is in the read state under four sets of conditions:

- The read state is active when the embedded flash memory is enabled (User Mode Read).
- The read state is active when MCR[PGM] or MCR[ERS] are high and high-voltage operation is ongoing (read-while-write).

Note: Reads done to the partition(s) being operated on (either erased or programmed) result in an error and the MCR[RWE] bit is set.

- The read state is active when MCR[PGM] and MCR[PSUS] are high (program suspend).
- The read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (erase suspend).

In embedded flash memory, flash core reads return 256 bits (1 page). Register reads return 32 bits of data.

Warning: Flash core reads are done through the BIU. In many cases the BIU does read page buffering to allow sequential reads to be done with higher performance. This could provide a data coherency issue that must be handled with software. Data coherency may be an issue after a program or erase operation, as well as UTest NVM block operations.

In user mode, registers may be written. Array may be written to do interlock writes.

Register reads to unmapped register address space return all zeroes.

Register writes to unmapped register address space have no effect.

Interlock writes that are attempted during a high-voltage operation (MCR[EHV] = 1 or MCR[DONE] = 0) will result in the interlock write being ignored, and address and data will not be updated.

Note: Care must be taken when doing 32-bit interlock writes. Performing two 32-bit interlock writes to the same word location is not supported. In addition, mixing 32-bit interlock writes with 64-bit interlock writes to the same double-word location is not supported. If these combinations are done, and a program operation is executed, the result will be MCR[PEG] = 0. See [Section 35.2.3.1, Module Configuration Register \(MCR\)](#) for more information.

35.3.1.2 Program

A flash memory program sequence operates on any page within the flash core. Within a page, up to eight words may be altered in a single program operation. Also, up to four pages can be altered in a single program operation. Whenever the array is programmed, the ECC bits also get programmed. ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed, since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be from 64 bits to 1024 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64-bit ECC segment between erases.

Warning: In rare cases over-programming of a 64-bit ECC segment may be done (EEPROM emulation).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

Note: *If a logic 0 is attempted to be over-programmed by a logic 1, the resulting operation will fail (MCR[PEG] = 0), and the 0's that are interlocked will be merged (ORed) with 0's that are already present in the 64-bit ECC segment, unless the block is designated at an Over-Program Protected block.*

Addresses in locked/disabled blocks cannot be programmed. Values may be programmed in any or all of eight words, within a page, with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. The program operation consists of the following sequence of events:

1. Change the value in the MCR[PGM] bit from a 0 to a 1.

Note: *Ensure the block that contains the address to be programmed is unlocked.*

2. Write the first address to be programmed with the program data. The embedded flash memory latches address bits [22:7] and SoC-specific UTest NVM enable at this time. The embedded flash memory latches data written as well. This write is referred to as a program data interlock write. An interlock write may be done as a 64-bit transaction or a 32-bit transaction. Refer to the BIU Block Guide for information on the size of writes that are allowed.
3. If more than one word or doubleword is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. The embedded flash memory ignores address bits [22:7] and SoC-specific UTest NVM enable for program data writes. All unwritten data words default to 0xFFFF_FFFF.
4. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step ⁽⁹⁾ to terminate.
5. Wait until the MCR[DONE] bit goes high.

Note: *Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.*

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more addresses are to be programmed, return to step ⁽²⁾.
9. Write a logic 0 to the MCR[PGM] bit to terminate the program sequence.

The first write after a program is initiated determines the quad-page address to be programmed. Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. The interlock write determines if the UTest NVM or normal array space is to be programmed by sampling SoC-specific UTest NVM enable and causing MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the values in MCR[PEAS] may be modified via the program interlock write, enabling erase-suspended programs to and from UTest NVM space.

An interlock write must be performed before setting MCR[EHV]. A program sequence may be terminated by clearing MCR[PGM] prior to setting MCR[EHV].

After the interlock write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024-bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of

0xFFFF_FFFF. If multiple writes are done to the same location, the data for the last write is used in programming.

While DONE is low, EHV is high, and PSUS is low, then EHV may be cleared, resulting in a program abort. A program abort forces the embedded flash memory to step eight of the program sequence. An aborted program results in PEG being set low, indicating a failed operation. The data space being operated on before the abort contains indeterminate data. A program sequence may not be aborted while in program suspend.

Warning: Aborting a program operation leaves the flash core addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

35.3.1.2.1 Program software locking

A software mechanism is provided to independently lock/unlock blocks against program and erase.

Software locking is done through the LOCK0, LOCK1, LOCK2, and LOCK3 registers. These may be written through register writes, and may be read through register reads.

35.3.1.2.2 Program hardware locking

Hardware locking which only affects the program operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

35.3.1.2.3 Program suspend/resume

The program sequence may be suspended to allow read access to the flash core. It is not possible to erase during a program suspend, or to program during a program suspend. These operations are prevented by register locking described in [Section 35.2.3.1, Module Configuration Register \(MCR\)](#). Read-while-write operation may also be used to read the array during a program sequence, providing the read is to a different partition.

A program suspend can be initiated by changing the value of the MCR[PSUS] bit from a 0 to a 1. MCR[PSUS] can be set high at any time when MCR[PGM] and MCR[EHV] are high. A 0 to 1 transition of MCR[PSUS] causes the embedded flash memory to start the sequence to enter program suspend, which is a read state. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended. At this time flash core reads may be attempted. Once suspended, the flash core may only be read. During suspend, all reads to flash core locations targeted for program, and blocks targeted for erase, return indeterminate data.

The program sequence is resumed by writing a logic 0 to MCR[PSUS]. MCR[EHV] must be set to a 1 before clearing MCR[PSUS] to resume operation. When the operation resumes, the embedded flash memory continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

Warning: Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory will

respond by completing the operation with a fail code (MCR[PEG] = 0).

35.3.1.2.4 Over-program protection enable

One-time programming can be enabled on a block basis. In an over-program protection enabled block, any doubleword which has already been programmed cannot be programmed again. The over-program protection enable does not affect erase operation. Attempts to over-program will result in MCR[PEG] being cleared. If any doubleword within the quad-page has an over-program protection violation, MCR[PEG] will be cleared, and no doublewords will be programmed.

One-time programming can be enabled for 16 KB, 32 KB, 64 KB, or 256 KB blocks.

Over-program protection is enabled by sideband signals, and if enabled, the program operation will be modified to check for programmed bits prior to doing a high-voltage program event.

35.3.1.2.5 Successful program address

To enable the SoC to create logic to authenticate operations with a program (such as in a diary operation), the flash module provides the last successful address programmed as a sideband signal. This value will be held until the next successful program. A successful program is defined as a program operation of data that is not all 1's to an unlocked block, and the operation results in MCR[PEG] returning a 1. If any doubleword within the pages being programmed is interlocked with all 1's, the sideband signal will not be updated. An address of all 1's, and each of the block size selects selected, is the default (and invalid) address after reset. This feature is valid for programs done on the main interface, and not valid for programs done on the alternate interface.

35.3.1.3 Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the main address space. The erase sequence is fully automated within the flash memory. Blocks to be erased must be selected prior to initiating the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest (low first, mid second, high last, 16 KB first, 32 KB second, 64 KB third, last blocks are 256 KB). The erase sequence consists of the following events:

1. Change the value in the MCR[ERS] bit from 0 to 1.
2. Select the block or blocks to be erased by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers.

Note: Lock and Select are independent. If a block is selected and locked, no erase occurs.

3. Write to any address in flash memory. This is referred to as an erase interlock write. An erase interlock write to UTest NVM space is not allowed if sealed (since erase is not allowed in the UTest NVM space when sealed).
4. Write a logic 1 to the MCR[EHV] bit to start an internal erase sequence or skip to step ⁽⁹⁾ to terminate.
5. Wait until the MCR[DONE] bit goes high.

Note: Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more blocks are to be erased, return to step ⁽²⁾.
9. Write a logic 0 to the MCR[ERS] bit to terminate the erase.

After setting ERS, one write, referred to as an interlock write, must be performed before EHV can be set to a 1. Data words written during erase sequence interlock writes are ignored (except for alternate erase interface). The erase sequence may be terminated by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV, assuming DONE is low, EHV is high, and ESUS is low. An erase abort forces the embedded flash memory to step eight of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. An erase sequence may not be aborted while in erase suspend.

35.3.1.3.1 Erase software locking

Software locking affects erase operation. For details on this see [Section 35.3.1.2.1, "Program software locking"](#).

35.3.1.3.2 Erase hardware locking

Hardware locking which only affects the erase operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

35.3.1.3.3 Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core. The erase sequence may also be suspended to program (Erase-Suspended Program) the flash core. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to flash core locations targeted for program and blocks targeted for erase return indeterminate data.

Read-while-write operation may also be used to read the array during an erase sequence, providing the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the embedded flash memory to start the sequence which places it in erase suspend. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended and further actions are attempted. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence, MCR[EHV] must first be cleared. If a program sequence is initiated, the values of SoC-specific UTest NVM enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its erase value. Flash core reads that occur while MCR[ESUS] = 1 from the block(s) being erased will return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase-suspended program)

before MCR[ESUS] can be cleared to resume the operation. The embedded flash memory continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

Warning: In an erase-suspended program, programming flash core locations in blocks which were being operated on will respond by completing the operation with a fail code (MCR[PEG] = 0). Programming voltages will not be applied to the memory array in this case.

Warning: Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory will respond by completing the operation with a fail code (MCR[PEG] = 0).

35.3.1.4 Alternate program and erase interface

A second interface is provided on the flash for program and erase operations. The intent is that a dedicated master can have private access to blocks through this interface.

The alternate interface includes an alternate MCR register and alternate lock registers. Any block in low, mid, or high address space may be accessible through the alternate interface. The UTest NVM Block is also accessible for program through the alternate interface. 256 KB sized blocks are not accessible through this interface.

The alternate interface supports programming of data sizes from one doubleword (64 bits) to four doublewords within a single page (256 bits). Within a page, up to eight words may be altered in a single program operation. It is recommended that programming operations through the alternate interface be from 64 bits to 256 bits, and be 64-bit aligned.

The alternate interface supports erasing a single block at a time with each erase operation.

As with the main interface, erase suspend, program suspend, and erase-suspended programs are supported.

Program and Erase times for the alternate interface are the same as the main interface when programs and erases are done in isolation. If main interface and alternate interface are being utilized simultaneously, bandwidth sharing is utilized.

Note: It is possible for the UTest NVM block to exist in both the main program interface and alternate program interface. In the event of a conflict the first interface to do the program to an address will get priority, and all other attempts after this will result in an OTP over-program failure (MCR[PEG] or MCRA[PEGa] being set to 0).

35.3.1.4.1 Alternate program interface

The programming procedure follows exactly the same flow as the main interface, except that only page programming is allowed. See section [35.3.1.2, "Program"](#) for more information on programming.

The alternate interface program is done at lower priority compared to the main interface. Based on the size of programming allowed, the main interface has a 4:1 ratio advantage over the alternate interface. Program operations are allowed to finish on the active interface prior to allowing the inactive interface to execute its operation. Suspend done on the active interface enable the inactive interface to begin its operation.

For alternate program while erase, there could be a latency of up to 5 ms in the program operation.

For more information on interaction between the alternate program interface and the main interface, see [Section 35.3.1.4.3, "Alternate interface performance"](#).

If the alternate interface is the only interface active, then normal program times may be expected.

35.3.1.4.2 Alternate erase interface

For erases done thru the alternate interface, the single block to be erased is selected based on the address that is received during the erase interlock write. Thus the Select registers are not valid for the alternate interface (step ⁽²⁾ of the erase flow in section [Section 35.3.1.3, Erase](#)), and the interlock write address is used (step ⁽³⁾ of the erase flow in section [Section 35.3.1.3, Erase](#)). All other features of the main interface apply to the alternate erase interface. The erase procedure is exactly the same as for the main interface. See section [Section 35.3.1.3, Erase](#) for more information on erase.

The alternate interface erase is done at lower priority compared to the main interface. The time-slice ratio between the two interfaces is 10:1, with 50 ms of time given to the main erase for every 5 ms of alternate interface. Suspend done on the active interface enable the inactive interface to begin its operation.

For alternate erase while program, there can be a latency of up to 5 ms in the main program time. Once the alternate erase is interrupted, it will remain interrupted for main programs for a fixed time of 50 ms, at which time the alternate erase will restart. It should be noted that multiple interruptions of the alternate erase will significantly increase the total erase time.

For alternate erase while erase, the main erase has priority and receives 50 ms of erase for every 5 ms of alternate erase. This may result in a significant increase in the erase time for the alternate interface, and a minor increase in erase time for the main interface.

For more information on interaction between the alternate erase interface and the main interface, see [Section 35.3.1.4.3, "Alternate interface performance"](#).

If the alternate interface is the only interface active, then normal erase times may be expected.

35.3.1.4.3 Alternate interface performance

[Table 491](#) highlights the interactions between the alternate interface and the main interface for program, erase, suspend, and idle conditions.

Table 491. Alternate program and erase characteristics

Alt PGM/ERS interface request	Main PGM/ERS interface request	Alt interface result on memory array	Main interface result on memory array
IDLE	IDLE	IDLE	IDLE
IDLE/SUS	PGM/ERS	IDLE/SUS	PGM/ERS
PGM/ERS	IDLE/SUS	PGM/ERS	IDLE/SUS
PGM	PGM	PGM	Priority PGM (up to 4:1)
PGM	ERS	PGM with latency of 5 ms	ERS (possible minimal increase to total ERS time)
ERS	PGM	ERS (significant increase in total ERS time expected)	Priority PGM (possible latency of 5 ms) Once PGM is granted, 50 ms time slice available for more programs
ERS	ERS	ERS (significant increase in total ERS time expected)	Priority ERS (10:1) (possible minor increase to total ERS time) 50 ms Main ERS for every 5 ms alternate ERS

35.3.2 Low Power mode

In Low Power mode, the embedded flash memory enables VDDF power gating, and disables circuits on the VFLASH domain. It is the lowest current mode for the flash memory. No reads from or writes to the embedded flash memory are possible when in Low Power mode. Most power dissipation is due to leakage in this mode.

Prior to entering Low Power Mode, it is required that all program, erase, and UTest operations be stopped prior to entering the mode. If Low Power mode is entered while an operation is suspended, entering Low Power mode will have the same effect as a reset during suspend.

When in Low Power mode, register access is prevented. Flash core accesses are also prevented until power mode is exited. Flash core reads and writes may occur after power mode is exited.

The embedded flash memory returns to a post-reset state in all cases.

35.3.3 UTest mode

UTest mode is a mode into which customers can put the embedded flash memory so as to do specific tests that check the integrity of the embedded flash memory.

35.3.3.1 Array integrity self-check

Array integrity is checked using a predefined address sequence (based on UT0[AIS]), and this operation is executed on selected blocks. The data to be read is customer-specific — user code may be programmed into the flash memory and the correct MISR signature is calculated based on that code. Any random or nonrandom code is valid. Once the operation is completed, the results of the reads can be checked by reading the MISR value, to determine if an incorrect read or ECC detection was noted. Array integrity MISR value is calculated after ECC detection and correction. Array integrity requires that the Read Wait States and Address Pipelined control registers in the BIU be set to match the system frequency being used. The array integrity check consists of the following sequence of events:

1. Enable UTest mode.
2. Select the block or blocks to be receive the array integrity check by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers. Blocks selected for array integrity check do not need to be unlocked.

Note: Unselected blocks are still read as part of the array integrity sequence, to enable full transition coverage. The result read on unselected blocks will not be captured in the MISR. Selecting fewer blocks may not result in a faster array integrity execution time, depending on the combinations of blocks selected and the sequence.

Note: Blocks protected with the Test Mode Disable Seal are still read as part of the array integrity sequence. The resulting read on sealed blocks will not be captured in the MISR, however the Single Bit Correction, Double Bit Detection and breakpoints will still be honored.

It is not possible to do array integrity operations on the UTest NVM block.

3. If desired, set the UT0[AIS] bit to 1 for sequential addressing only.

Note: For normal integrity checks of the flash memory, sequential addressing is recommended. If it is required to more fully check the read path (in a diagnostic mode), it is recommended that AIS be left at 0, to use the address sequence that checks the read path more fully, and examine read transitions. This sequence takes more time.

4. Seed the MISR UM0 thru UM9 with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, ensure that UT0[SBCE] is set.
6. Set the UT0[AIE] bit.
 - i) If desired, the array integrity operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and then continuing to the next step. It should be noted that in the event of an aborted array integrity check the MISR registers will contain a signature for the portion of the operation that was completed prior to the abort, and will not be deterministic. Prior to doing another array integrity operation, the UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8, and UM9 registers may need to be initialized to the desired seed value by doing register writes.
 - j) If desired, the array integrity operation may be suspended prior to UT0[AID] going high. UT0[AISUS] may be set to request an array integrity suspend. After the UT0[AISUS] bit is set, the user should wait for UT0[AID] bit to go high, which indicates the flash has entered the suspend state, and normal reads to the flash may be done. Once [AID] goes high, UT0[AISUS] may be cleared to resume the array integrity sequence.

Note: User mode array reads requested during the array integrity test will be ignored, to ensure that the array integrity operation is not corrupted. The memory array will not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

7. Wait until the UT0[AID] bit goes high.
8. If breakpoints were enabled, check UT0[NAIBP], and if this is set to 1, the MCR[EER], MCR[SBC], and ADDR registers may be checked to determine the cause of the break and the address of the break. Prior to resuming the operation, clear MCR[SBC] or MCR[EER] bits. Then the operation may be resumed by clearing UT0[NAIBP]. Continue to wait until the UT0[AID] bit goes high. If breakpoints were not enabled, or if UT0[NAIBP] is low when UT0[AID] goes high, then the operation is complete. Continue to the next step.
9. Read values in the MISR registers (UM0 through UM9) to ensure correct signature.

Note: Array integrity reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

10. Write a logic 0 to the UT0[AIE] bit.

35.3.3.2 User margin read

User margin read may be done using the array integrity interface, and has all the associated features of the array integrity interface (MISR and Breakpoints). User margin reads are done at a read margin level checking for erased bits or programmed bits encroaching on the nominal read level. User margin read is a self-timed event, and is independent of system clocks or wait states selected. Margin ECC corrections and detections are noted during the user margin read test. Margin read MISR value is calculated after ECC detection and correction.

The data to be read is customer-specific, and user code may be programmed into the flash memory. Any random or non-random code is valid. Once the operation is completed, the margin read results can be checked by reading the MCR[EER] bit and the MCR[SBC] bits to determine if zero, one, or two bits are being detected by the margin read, as well as checking the MISR.

The use model for margin read is in the event of a user-detected single bit correction (through user reads). A margin read may be done to check for a possible second bit falling within the selected margin levels.

The procedure for doing margin reads is:

1. Enable UTest mode.
2. Select the block or blocks to receive margin read check by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers. Blocks selected for margin read do not need to be unlocked.

Note: Unselected blocks are still read as part of the margin read sequence, to enable full transition coverage. The result read on unselected blocks will not be captured in the MISR. Selecting fewer blocks will not result in a faster margin read execution time.

Note: Blocks protected with the Test Mode Disable Seal are still read as part of the margin read sequence. The resulting read on sealed blocks will not be captured in the MISR, however the Single Bit Correction, Double Bit Detection and breakpoints will still be honored.

Note: It is not possible to do margin read operations on the UTest NVM block.

3. Set the UT0[AIS] bit to 1 for sequential addressing only.

Note: For margin read checks of the flash memory, sequential addressing is recommended. Setting AIS to 0 is possible for Margin Reads, but using the sequence takes more time, and is not recommended.

4. Seed the MISR UM0 thru UM9 with desired values.
5. Ensure that the MCR[EER] and MCR[SBC] bits are cleared.
6. If you want to detect single bits during margin read, set UT0[SBCE] to 1.
7. Set the UT0[MRE] bit.
8. Set the UT0[MRV] bit to the desired value depending on if it is desired to do one's margin or zero's margin.
9. Set the UT0[AIE] bit.

Note: User mode array reads requested during the margin read test will be ignored, to ensure that the margin read operation is not corrupted. The memory array will not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

Note: During Margin Read operations, with AID low, it is not recommended to attempt write operations to the MCR[SBC] and MCR[EER] bits. It is recommended that these bits only be written during suspend, breakpoints, or at completion of the Margin Read operation.

10. Wait until the UT0[AID] bit goes high.
 - a) If breakpoints were enabled or a suspend was requested during margin read, the operation may be at a breakpoint or a suspend state. See [Section 35.3.3.1, "Array integrity self-check"](#) for more information.
11. Read values in the MISR registers (UM0 through UM9) to ensure correct signature.

Note: Margin reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

12. Write a logic 0 to the UT0[AIE] bit.

35.4 Initialization information

A reset is the highest priority operation for the embedded flash memory and terminates all other operations.

The embedded flash memory uses reset to initialize register and status bits to their default reset values. If the embedded flash memory is executing a program or erase operation (PGM or ERS = 1) and a reset is issued, the operation is aborted and the embedded flash memory disables the high-voltage logic without damage to the high-voltage circuits. Reset aborts all operations and forces the embedded flash memory into user mode ready to receive accesses.

After reset is requested, MCR[DONE] goes low, and remains low during reset and reset recovery. At the end of reset recovery, MCR[DONE] transitions from a 0 to a 1.

After reset is completed, register reads may be done, although it should be noted that registers that require updating from UTest NVM information, or other inputs, may not read updated values until MCR[DONE] transitions high.

During reset recovery, register writes are not allowed until the MCR[DONE] bit transitions high to indicate reset recovery is completed.

Warning: Resetting during a program or erase operation leaves the flash core blocks being programmed or erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

36 Decorated Storage Memory Controller (DSMC)

36.1 Introduction

This section details hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called “*decorated storage*”. It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A “decoration” is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l [b, h, w] d{cb}x rT, rB, rA` where the data size specifier is defined as 8-bit (*b* = byte), 16-bit (*h* = halfword) or 32-bit (*w* = word), and the three register specifiers include *rT* as the destination target register, *rB* as the effective address and *rA* as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st [b, h, w] d{cb}x rS, rB, rA` where the same data size specifiers are used, and the three register specifiers are *rS* as the source data register, *rB* as the effective address and *rA* as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the *rB* register (hence the “*x*” mnemonic suffix, signaling an “indexed” addressing mode) and the decoration is specified by the *rA* register. The core transmits the contents of the *rA* register as the decoration value along with the access address (register *rB*) to the DSMC.

The decorated load and store instruction formats are shown in [Figure 421](#).

Figure 421. Power Architecture Decorated Load and Store Instruction Formats

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1			rT				rA				rB				1	0	0	0	0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1			rT				rA				rB				1	0	0	0	1	0	0	0	0	1	1	/
lwdx	0	1	1	1	1	1			rT				rA				rB				1	0	0	1	0	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1			rT				rA				rB				1	0	0	0	0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1			rT				rA				rB				1	0	0	0	1	0	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1			rT				rA				rB				1	0	0	1	0	0	0	0	0	1	0	/
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
stbdcx	0	1	1	1	1	1			rS				rA				rB				1	0	1	0	0	0	0	0	0	1	1	/
sthdcx	0	1	1	1	1	1			rS				rA				rB				1	0	1	0	1	0	0	0	0	1	1	/
stwdx	0	1	1	1	1	1			rS				rA				rB				1	0	1	1	0	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1			rS				rA				rB				1	0	1	0	0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1			rS				rA				rB				1	0	1	0	1	0	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1			rS				rA				rB				1	0	1	1	0	0	0	0	0	1	0	/

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit “decoration”. The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and/or operand data for the command.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (l [b, h, w] dcbx) and store (st [b, h, w] dcbx) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

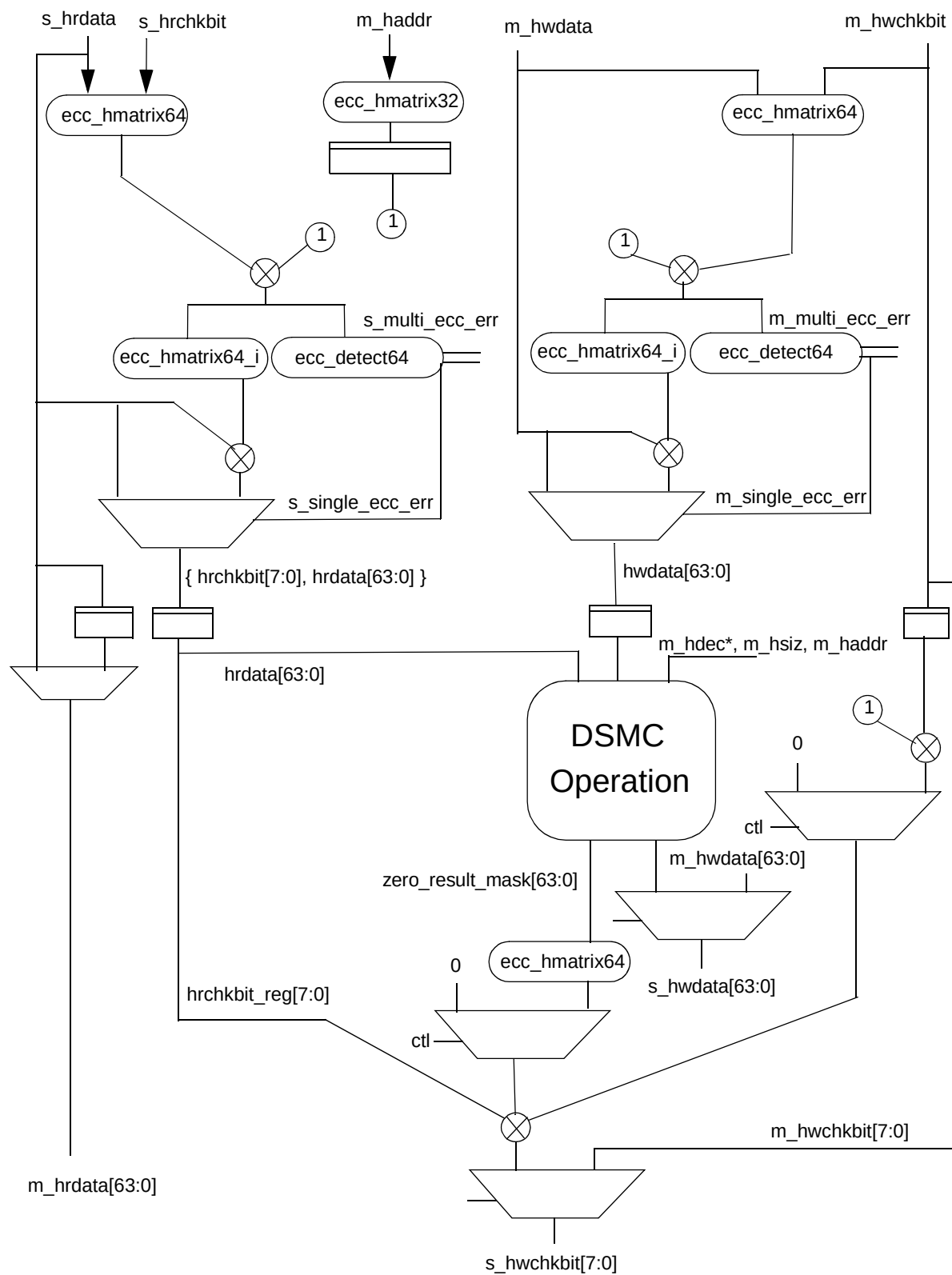
The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

36.2 DSMC Block Diagram

The following block diagram, [Figure 422](#), shows the basic datapaths for the ECC logic in the DSMC. On both the read data path (hrdata) and the write data path (hwdata) there is logic to detect an ECC error and correct the data. The circle with X inside represents an XOR operation. The ecc_hmatrix and ecc_detect blocks are coefficient matrixes based on the Hsiao ECC algorithm. Notice in the recalculation of the s_hwchckbit[7:0] that there is a read chckbit from the slave (hrchckbit), a zero padded result contribution, and a contribution from the hwchckbit from the master. The details of the DSMC bit field and other operations are not shown (represented by the DSMC Operation box).

Figure 422. DSMC Block Diagram



36.3 Decorated Stores: `st [b,h,w] d{cb}x rS, rB, rA`

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a `<command>. <size>` syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- wdata = rS

36.3.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit Memory Container

Figure 423. Decoration Format: BFINS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bfins.b	0	0	0	0	0	0	0	SRTBIT	0	0	0	BFW	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
bfins.h	0	0	0	0	0	0	SRTBIT			0	0	BFW			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
bfins.w	0	0	0	0	0	SRTBIT				0	BFW					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

This command inserts a bit field defined by SRTBIT and BFW into the memory “container” defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read

if bfw == 0
    then bfw = container

if ((srtbit + bfw) <= container) // generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp & ~mask // modify
      | wdata & mask

mem[accessAddress, size] = tmp // memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field “xyz” into an 8-bit memory container, initially set to “abcd_efgh”. For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz_----,
    then destination is "xyzd_efgh"
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
    then destination is "axyz_efgh"
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
    then destination is "abxy_zfgh"
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
    then destination is "abcx_yzgh"
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
    then destination is "abcd_xyzh"
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_xyz,
    then destination is "abcd_xyz"
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
    then destination is "abcd_efxy"
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
    then destination is "abcd_efgx"
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

36.3.1.1 Compare-and-Store (CAST)

Figure 424. Decoration Format: CAST

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cast.b	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CD8							
cast.h	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	CD16															
cast.w	1	0	0	1	CD28																											

This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size] // memory read

if (size == 8) // define compare_operand
    compare_operand = CD8
else if (size == 16)
    compare_operand = CD16
    else compare_operand = {0x0, CD28} // zero-filled data

if (tmp == compare_operand) // compare - "modify"
    mem[accessAddress, size] = wdata // memory write
else mem[accessAddress, size] = tmp
```

Note for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit compare operand.

36.3.1.1.1 Logical AND (AND)

Figure 425. Decoration Format: AND

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
and. {b,h,w}	1	0	1	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata // modify
mem[accessAddress, size] = tmp // memory write
```

36.3.1.1.2 Logical OR (OR)

Figure 426. Decoration Format: OR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
or. {b,h,w}	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata // modify
mem[accessAddress, size] = tmp // memory write
```

36.3.1.1.3 Logical Exclusive-OR (XOR)

Figure 427. Decoration Format: XOR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
xor. {b,h,w}	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata // modify
mem[accessAddress, size] = tmp // memory write
```

36.3.1.2 Decorated Loads: 1 [b, h, w] d {cb} x rT, rB, rA

The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an “extract”) operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

36.3.1.2.1 Simple Load (SLD)

Figure 428. Decoration Format: SLD

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sld. {b,h,w}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

36.3.1.2.2 Registers-and-Memory Exchange (SWAP)

Figure 429. Decoration Format: SWAP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cast.b	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	WD8							
cast.h	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	WD16															
cast.w	0	1	0	1	WD28																											

This command loads the contents of the referenced memory location into the core's rT register and stores the source operand from the core's rA register (the decoration) into the memory location.

```
tmp = mem[accessAddress, size] // memory read
```

```
rdata = tmp // rT = rdata = tmp
```

```
if (size == 8) // store_operand - "modify"
```

```
    store_operand = WD8
```

```
else if (size == 16)
```

```
    store_operand = WD16
```

```
    else store_operand = {0x0, WD28} // zero-filled write data
```

```
mem[accessAddress, size] = store_operand // memory write
```

Note for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit store operand.

36.3.1.2.3 Load-and-Set-1(Bit) (LAS1)

Figure 430. Decoration Format: LAS1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	0	BIT		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp  = mem[accessAddress, size] // memory read
rdata = tmp // rT = rdata = tmp
mask  = 1 << (container - bit - 1) // generate bit mask
tmp   = tmp | mask // modify
mem[accessAddress, size] = tmp // memory write
```

36.4 DSMC Timing Diagram

The following timing diagram, [Figure 431](#), shows a decorated load followed by a decorated store instruction which is then followed by a normal load instruction. Both decorated instructions are read-modify-write instructions and thus they take at least two cycles to execute. In the first cycle an ECC calculation is performed on the read data as well as the write data. Then in the second cycle of the decorated instruction the decorated operation is performed and if it is a decorated store, the result is merged with the read data and sent to the write data bus along with the recalculated ECC write chkbits.

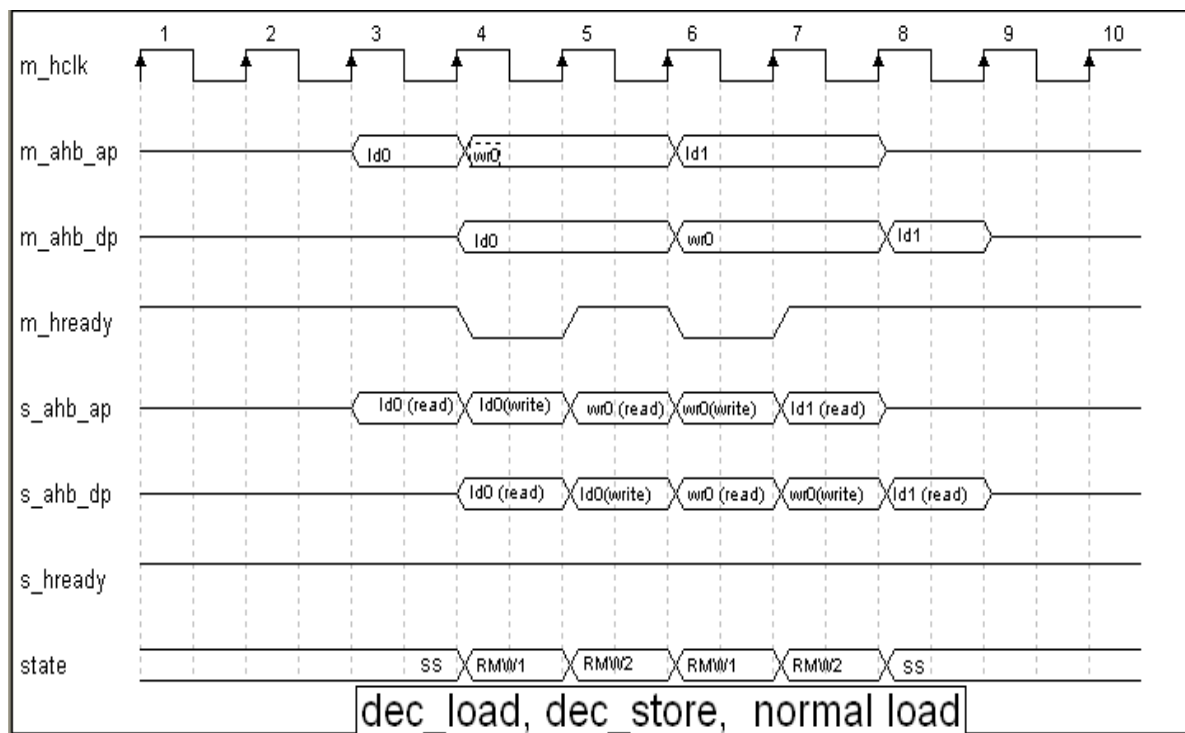


Figure 431. DSMC Timing Diagram

36.5 DSMC Instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated memory controller only operates on aligned, single data transfers (misalignment and/or bursts are not supported and error terminated if attempted).

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

37 Flash Memory Programming and Configuration

This chapter provides basic guidance on activities related to configuring and programming the embedded flash memory in the SPC57EM80xx microcontroller.

Multiple layers of protection are available to prevent unintended updates to flash contents and unauthorized reads and writes to flash contents. The protection mechanisms range from simple write-protect bits for individual flash blocks to four levels of 256-bit password read and/or write protection and configuration of flash memory blocks as One Time Programmable (OTP).

Additionally, the SPC57EM80xx microcontroller includes a tamper detection feature, debug port enable/disable, and the ability to disable the ability for the device to be put into factory test mode.

Factory test mode disable
Configurable One Time Programmable (OTP) flash protection
Configurable tamper detection (erase counter)
Debug port enable/disable
Secure read protection (one to four 256-bit passwords)
Configurable secure write protection (one to four 256-bit passwords)
non-secure write protection
Flash blocks must be explicitly selected for erase operations
Code and data

Figure 432. Flash memory protection layers

At the lowest layer of protection, all flash erase operations require the explicit selection of one or more blocks of flash memory before the operation is performed. When executed, the operation is only performed on the selected block(s).

The next layer is the chip's ability to lock individual flash blocks against unintended programming and erase operations. This level of protection does not offer any password protection and is intended to provide protection against *accidental* changes to code and data.

The first layer of *secure* protection is the microcontroller's ability to support up to four levels of 256-bit password-secured *write* protection for individual flash blocks.

Similarly, up to four levels of password-secured *read* protection can be implemented. Unlike the password-secured write protection, which is implemented on a per-block basis, password-secured read protection applies to five pre-defined (at the factory) groups of flash blocks. The read protection applied to a group applies to each block within the group. In contrast, secure write protection is defined at the individual block level.

Secure password protection can also be implemented for the debug port.

The SPC57EM80xx microcontroller also includes a tamper detection feature. In its most basic implementation, tamper detection acts as an erase counter. If tamper detection is implemented, a DCF record in the tamper detection *diary* must be written by software before a protected flash block can be erased. Multiple tamper detection regions can be defined by the customer, with each region containing one or more blocks. A block can be included in more than one tamper detection regions if desired.

Related to tamper detection is the ability to configure any flash block as OTP, so it can only be programmed once.

Finally, the microcontroller includes a feature that enables the customer to disable the device's ability to be placed in factory test mode.

The remainder of the chapter provides details these protection mechanisms and gives a brief overview of security configuration planning.

Note: Most of the topics in this chapter build on information presented in previous topics. To get a clear overview of how the flash protection features work and relate to each other, please read the chapter in sequence, from beginning to end.

With the exception of read protection and tamper detection, which are applied to groups of flash memory blocks, access protection configuration is applied on a per-block basis, i.e., a custom level of access protection can be provided for each flash memory block. This is accomplished through the use of registers containing bits that are mapped to individual flash blocks. Understanding the mapping is key to understanding the implementation and use of the SPC57EM80xx microcontroller's flash protection features.

Depending on the type of protection to be applied, different sets of registers (or even DCF records) are used, but the mapping scheme is consistent. This mapping scheme is first presented in [Section 37.1, Selection of flash memory blocks for erase](#). It is important to understand the topic before proceeding to other sections.

37.1 Selection of flash memory blocks for erase

The registers used for selecting flash blocks for erase operations are the flash control registers named LOCK0, LOCK1, and LOCK2. There is also a LOCK3 register, but no flash blocks are mapped to it.

The LOCK0–2 registers are 32-bit registers containing fields associated to a specific address space: low address space, mid address space, high address space, or 256 KB address space. Each non-reserved bit in the fields is mapped to a specific flash block.

For example, [Figure 433](#) shows the flash module's Select 0 (SEL0) register. This register provides a way to select blocks in the low- and mid address spaces for erase operations.

Figure 433. SEL0 flash control register

Offset 0x0038												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWSEL[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDSEL[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 492 shows an example of a mapping of SEL0 register's LOWSEL and MIDSEL field bits to individual flash blocks⁽ⁱ⁾. In this case, not all bits of the LOWSEL and MIDSEL fields are mapped—only LOWSEL bits 0–11 (register bits 15–4^(j)) and MIDSEL bits 0–2 are mapped. Before any mapped block can be erased, it must be selected by writing a '1' to its corresponding bit in one of the SEL n registers. See the device configuration chapter for the complete mapping.

Note: Values written to the SEL registers do not persist through device reset. As indicated by the reset values shown in Figure 433, all bits are reset to '0'. As a result, after a device reset all blocks mapped to the register are, by default, not selected for erase.

Table 492. Sample low and mid address space flash block bit mapping

Start Address	End Address	Size (KB)	Partition	LOCK n (1) Register	LOCK n Register Field Bit	LOCK n DCF Record DATA Bit ⁽²⁾	Read Lock group (3)	Block No.	Description
0x0040_4000	0x0040_7FFF	16	0	LOCK0	LOWLOCK[0]	16	—	0	BAF ⁽⁴⁾
0x00FC_0000	0x00FC_3FFF	16	0	LOCK0	LOWLOCK[1]	17	1	1	Code Flash ⁽⁵⁾
0x00FC_4000	0x00FC_7FFF	16	0	LOCK0	LOWLOCK[2]	18	1	2	Code Flash ⁴
0x00FC_8000	0x00FC_BFFF	16	1	LOCK0	LOWLOCK[3]	19	1	3	Code Flash ⁴
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash ⁴
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

i. The table lists the LOWLOCK register field but the structure is identical to the LOWSEL field in Figure 433.

j. Register bits in Power Architecture-based microcontrollers are numbered with the most significant bit as 0. This is not necessarily the case with register field bit numbering. The field bit numbering shown in the figure has the least significant bit numbered as 0.

Table 492. Sample low and mid address space flash block bit mapping (continued)

Start Address	End Address	Size (KB)	Partition	LOCK _n (¹) Register	LOCK _n Register Field Bit	LOCK _n DCF Record DATA Bit(²)	Read Lock group (³)	Block No.	Description
0x00FD_0000	0x00FD_7FFF	32	0	LOCK0	LOWLOCK[6]	22	1	6	Code Flash
0x00FD_8000	0x00FD_FFFF	32	1	LOCK0	LOWLOCK[7]	23	1	7	Code Flash
0x00FE_0000	0x00FE_FFFF	64	0	LOCK0	LOWLOCK[8]	24	1	8	Code Flash
0x00FF_0000	0x00FF_FFFF	64	0	LOCK0	LOWLOCK[9]	25	1	9	Code Flash
0x0061_0000	0x0061_FFFF	64	1	LOCK0	LOWLOCK[10]	26	3	10	Secure Code
0x0062_0000	0x0062_FFFF	64	1	LOCK0	LOWLOCK[11]	27	3	11	Secure Code
—	—	—	—	—	—	—	—	—	—
0x0068_0000	0x0068_3FFF	16	2	LOCK0	MIDLOCK[0]	0	4	12	Secure Data
0x0068_4000	0x0068_7FFF	16	3	LOCK0	MIDLOCK[1]	1	4	13	Secure Data
—	—	—	—	—	—	—	—	—	—
0x0040_0000	0x0040_3FFF	16	0	LOCK0	TSLOCK	31	0	32	UTEST(⁶)

1. There are multiple sets of LOCK_n registers, i.e., the flash module's LOCK0—3 registers and PASS module's PASS_LOCK0—3_PG0—3, but all use the same mapping of register bits to flash blocks. The same mapping applies the flash module's SEL0—3 registers.
2. The DCF record DATA bit refers to the DATA[31:0] field contained in DCF records created during OEM configuration activities. The DCF records related to flash blocks are used to implement password-secured write protection, password-secured read protection, tamper detection, and debug port enable/disable as a permanent part of the device configuration.
3. The Read Lock group pertains to protecting code and data blocks from being read by debuggers. This type of protection is implemented using DCF records and the PASS_LOCK3_PG0—3 registers.
4. The Boot Assist Flash (BAF) is one time programmable and the majority of it is programmed in the factory. The BAF handles the initial start up of the device.
5. This block is searched to determine the boot location (if it contains the boot header).
6. One time programmable.

Example 6 Selecting blocks for erase

Referring to [Table 492](#), writing the value 0x0030_0000 to the SEL0 register ([Figure 433](#)) selects the following blocks for erase:

Start Address	End Address	Size (KB)	Partition	LOCK _n Register	LOCK _n Register Field Bit	LOCK _n DCF Record DATA Bit	Read Lock group	Block No.	Description
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash ⁴
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

Note: The mappings of SEL_n register bits to flash blocks vary with different microcontrollers. See the flash memory section in the Device Configuration chapter of this reference manual for the mappings specific to the SPC57EM80xx microcontroller.

37.2 Non-secure write protection

Non-secure write protection refers to the chip's ability to lock individual flash blocks against programming and erase operations. This level of protection is intended to provide protection against *accidental* changes to code and data. Although similar in function, this is *not* associated with the built-in register protection (REG_PROT) feature included with many of the chip's modules. The non-secure write protection feature detailed here applies only to embedded flash memory.

Implementing non-secure write protection is almost identical to selecting specific flash blocks for an erase operation—the only difference is the register used. To provide non-secure write protection for a block, write a '1' to the corresponding mapped bit found in one of the LOCK0–2 flash control registers.

Note: As indicated by the reset values shown in [Figure 434](#), after reset all blocks mapped to the register are, by default locked against programming and erase. Values written to the LOCK and SEL registers do not persist through device reset.

The same bit mapping that applies to the SEL0–3 registers also applies to the LOCK0–3 registers.

Offset 0x0010												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCK	0	LOWLOCK													
W																
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCK															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Referring to [Table 492](#), writing the value 0x0030_0000 (same value used in [Example 6](#)) to the LOCK0 register ([Figure 434](#)) locks the following blocks (same blocks selected in [Example 6](#)) so they cannot be programmed or erased, even if selected for erase:

Start Address	End Address	Size (KB)	Partition	LOCK <i>n</i> Register	LOCK <i>n</i> Register Field Bit	LOCK <i>n</i> DCF Record DATA Bit	Read Lock group	Block No.	Description
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash ⁴
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

As long as the LOCK0[LOWLOCK] bits are unchanged, the above blocks cannot be modified unless the protection is overridden (discussed in [Section 37.3, Secure write protection](#)). All other blocks mapped to bits in the LOCK0 *can* be programmed because their corresponding bits have been set to '0'.

Secure write protection refers to the chip's ability to implement up to four levels of 256-bit password-secured *write* protection for individual flash blocks. Use of this feature is optional but the default behavior of the SPC57EM80xx microcontroller is to apply secure write protection. If there is a need to *not* write-protect specific blocks, that needs to be configured by creating the appropriate DCF records.

Note that the *secure* write protection settings override the non-secure write protection detailed in [Section 37.2, Non-secure write protection](#).

Implementing a custom secure write protection scheme requires creating DCF records that define the level of write protection to be applied to each flash block. The DCF records use a bit mapping scheme identical to the mapping in the flash module's `SELn` and `LOCKn` registers. The write protection scheme becomes a permanent part of the device but blocks can be temporarily unlocked. Depending on the specific implementation, temporarily unlocking the write protection requires matching up to four 256-bit passwords.

The easiest way to understand the secure write protection functionality is to examine the role played by the `PASS_LOCK0_PGn`–`PASS_LOCK3_PGn` registers.

The `PASS_LOCK0_PGn`–`PASS_LOCK3_PGn` registers are four identical sets of registers contained in the PASS module—they distinguished by the `PGn` suffix. The `PGn` (Password Group *n*) suffix indicates the password required to unlock that set of registers. Password 0 unlocks `PASS_LOCK0_PG0`, `PASS_LOCK1_PG0`, `PASS_LOCK2_PG0`, and `PASS_LOCK3_PG0`. Similarly, the registers suffixed with `PG1` are unlocked by matching password 1, and so on.

The `PASS_LOCK0_PGn`–`PASS_LOCK3_PGn` register sets have two functions related to secure write protection:

- Reading the registers provides the secure write lock status for each flash block.
- Changing the value of a mapped bit in a register changes the secure write protect status of the corresponding flash block.

Each set of registers contains the same mapping of bits to flash blocks as the flash module's `SELn` and `LOCKn` registers. In any of the four `PASS_LOCKx_PGn` register sets, if a mapped bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access to the register, and setting the bit to '0'. If the same bit is set to '1' in more than one `PASS_LOCKx_PGn` register set, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

[Figure 435](#) shows how the registers interact. Note that secure write protection is only affected by the settings in `PASS_LOCK0_PGn`–`PASS_LOCK2_PGn`. The `PASS_LOCK3_PGn` registers have additional security functions, discussed in later topics.

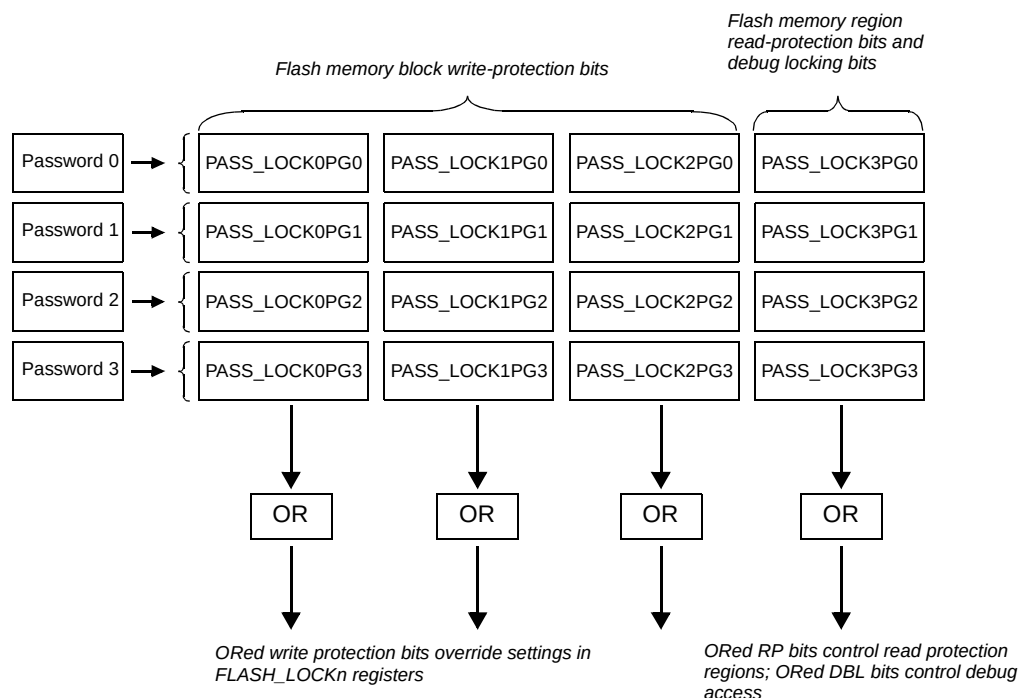


Figure 435. Secure write protection

The next sections discuss implementing secure write protection and overriding secure write protection.

37.3.1 Implementing secure write protection

Implementing secure write protection is a one-time task performed either at the factory or by the customer during chip configuration activities. It normally requires creating a DCF record for each of the PASS_LOCK0_PGn–PASS_LOCK2_PGn registers^(k). The DCF records have a structure that includes the same mapping of bits to flash memory blocks as the flash module's SELn and LOCKn registers. The values assigned become the permanent register reset values for the PASS_LOCK0_PGn–PASS_LOCK2_PGn registers.

See [Chapter 9: Device Configuration Format \(DCF\) Records](#) for complete details on DCF records. This section summarizes the contents of a DCF record as required to implement secure write protection.

Even though the mapping of bits to flash memory blocks is identical to the mapping used for selecting flash blocks for erase or non-secure write locking, there are two significant differences:

k. The default behavior for secure write protection in the case where passwords are created but no secure write protection DCF records are created, is for all blocks to have four levels of password-secured write protection. To vary the protection among all blocks, a DCF record for each password group must be created.

1. Configuring flash blocks for secure write protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. Configuration of blocks for secure write protection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

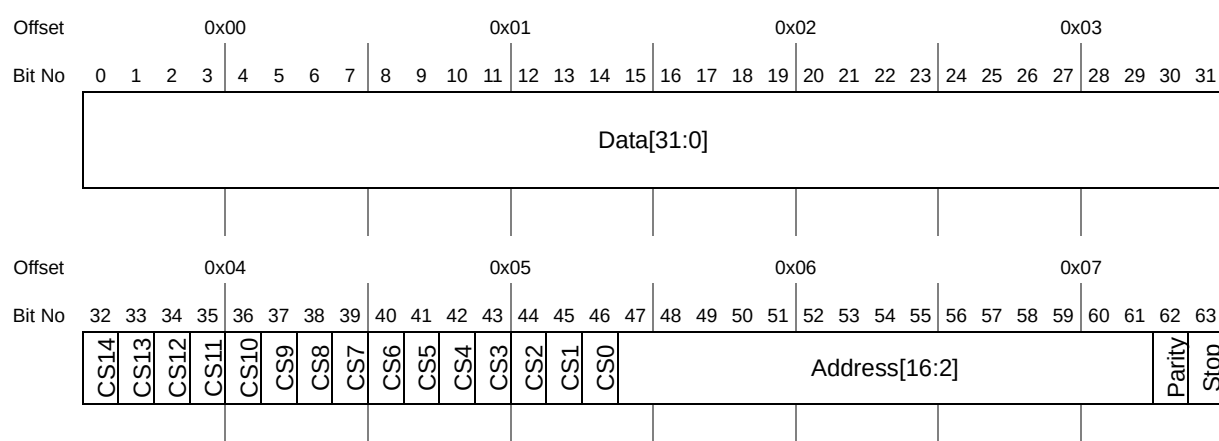
Caution: Careful planning is required before writing DCF records. They are created in a write-once area of flash. Multiple instances of any DCF record can be created, but in some cases only the record created first is used and with others only the most-recently written record is used. Be sure to understand the characteristics of a DCF record before creating it.

The DCF records required for implementing a secure write protection scheme are detailed in the PASS section of DCF client list table ([Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#)). Table entries provide the CS and Address value combinations that correspond to PASS_LOCK0_PGn–PASS_LOCK3_PGn registers.

The format of a secure write protection record is summarized here for convenience.

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTest flash memory. The structure of a DCF Record is shown in [Figure 436](#).

Figure 436. DCF record structure



Note: The values in the PASS_LOCKxPGn registers override the settings in the flash module's LOCK0–3 registers.

Values written to the PASS_LOCKxPGn registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase.

The PASS_LOCKxPGn bits take effect when the life cycle is matured to "OEM Production" or older.

Example 8 Implementing secure write protection

Implementation of secure write protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) must be stored in a protected OTP area of device flash called UTest flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PGn–PASS_LOCK3_PGn) that have the same mapping of

register bits to flash blocks as the flash module's SEL0–3 registers and LOCK0–3 registers.

- DCF records must be created to define which flash memory blocks are to be locked and which blocks are not locked.

The records defined in step ⁽²⁾, determine the reset values of the PASS_LOCK0_PGn–PASS_LOCK3_PGn registers and therefore the write protection settings that take effect each time the microcontroller is reset.

Assume four passwords have been previously implemented and the following blocks are to be locked in the password 0 group:

Start Address	End Address	Size (KB)	Partition	LOCKn Register	LOCKn Register Field Bit	LOCKn DCF Record DATA Bit	Read Lock group	Block No.	Description
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash ⁴
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

From the PASS section of [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#):

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0000	LOCK0_PG0

Referring to [Figure 436](#), the values for the DCF record are:

- Data[31:0] is 0x0030_0000 (bits 0–31) (same value used in previous examples)
- CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
- Address[16:2] is 0b000_0000_0100_0000 (bits 47–61)
- Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
- Preferred Stop value is 0 (see footnotes in [Figure 437](#)).

The following DCF record provides secure write protection for the blocks in [Table 492](#) by setting the reset values PASS_LOCK0_PG0 to the appropriate value to lock the blocks.

Figure 437. Sample DCF record for secure write protection

Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit No	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	X (1)	X (2)

1. Bit 62 indicates parity strategy, which is not used in the PASS DCF records so it is a "don't care".
2. Bit 63 indicates whether the current record is the last DCF. Generally, this should be written as '0' so the first unprogrammed (all 1's) DCF record acts as the stop record.

The remaining blocks mapped to the PASS_LOCK0_PGn registers do *not* have secure write protection provided by the password 0 register set because the DATA field of DCF record has only the bits mapped to flash blocks 4 and 5 set to '1'.

Caution: The above example is intended to highlight the need to carefully understand how DCF records work. The protected blocks are secured by password 0 but by default they are also secured by passwords 1, 2, and 3. In contrast the blocks *not* secured by password 0 in this example *are* secured by passwords 1, 2, and 3.

See the PASS section of DCF client list table ([Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#)) for details on the DCF records.

37.3.2 Overriding secure write protection

The "PGn" (Password Group *n*) suffix on each the register names indicates the password each set of registers is associated with. For example, password 0, is used to unlock PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. The bits in the first three registers (PASS_LOCK0_PG0, PASS_LOCK1_PG0, and PASS_LOCK2_PG0) are mapped exactly the same as the bits in the flash module's SEL0–2 registers and LOCK0–2 registers.

Like the SEL0 and LOCK0 registers, the PASS_LOCK0_PGn registers all contain bits mapped to the low and mid address spaces, and the mapping scheme is the same. A mapped bit set to '1' in any of the registers causes the corresponding block to be locked against program or erase. A mapped bit set to '0' in any of the registers makes the corresponding block available for programming or erase.

If a bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access, and setting the bit to '0'. If the same bit is set to '1' in more than one of the PASS_LOCKxPGn register sets, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

[Figure 438](#) shows the PASS_LOCK0_PGn register. As you can see, it has the same structure as the flash module's LOCK0 register. In addition to the main difference of being among a group of four parallel sets of locking registers, the other difference between the PASS_LOCKx_PGn registers and the flash LOCKn registers is how the reset values are determined.

By default, all mapped bits are set to '1', locking their corresponding flash blocks from programming and erase. The reset value is user-configurable, though, by writing a DCF record. This is discussed in a later section.

Figure 438. LOCK0_PGn register

0x0100 LOCK0_PG0																	
Offset 0x0110 LOCK0_PG1																	
0x0120 LOCK0_PG2																	
0x0130 LOCK0_PG3																	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		TSLOCK		ATSL		LOWLOCK											
W																	
Reset		x ¹	0	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		MIDLOCK															
W																	
Reset		x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ⁽¹⁾

1. If no DCF records have been written to change the lock state, the bit will reset to '1'

Note: The values in the PASS_LOCKxPGn registers override the settings in the flash module's LOCK0–3 registers.

Values written to the PASS_LOCKxPGn registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase. The purpose of the registers is to indicate secure write protection status and to provide a way to temporarily override secure write protection. The PASS_LOCKxPGn bits take effect after the life cycle is matured to "OEM Production" or older.

Example 9 Overriding secure write protection

Assume four passwords have been created but no DCF records have been written to set the reset values of the PASS_LOCKx_PGn registers, so the reset value for each mapped bit defaults to '1', causing all mapped blocks to have four levels of 256-bit password-secured write protection.

Since the settings in the PASS_LOCKx_PGn registers override the settings in the flash module's LOCK0–3 registers, the settings in those registers are irrelevant. Each of the four sets of the PASS module's LOCK registers must be unlocked by supplying the correct password for each set and then the appropriate bits must be set to '0' to make the desired flash blocks available for program or erase.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CINn) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write 0xFFCF_FFFF to the PASS_LOCK0_PG0 register and verify a successful write.
4. Repeat the above steps for passwords 1–3.

Just as in previous examples, the result is the following blocks are available for program and erase operations but at a much higher level of security.

Start Address	End Address	Size (KB)	Partition	LOCK n Register	LOCK n Register Field Bit	LOCK n DCF Record DATA Bit	Read Lock group	Block No.	Description
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash ⁴
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

37.4 Secure read protection

Read Protection is ability of the flash controller to block read accesses to the flash memory from all bus masters. This includes the blocking of:

- Instruction fetches and load instructions from the CPU
- Debug reads from an outside tool
- DMA reads
- SIPI reads
- All other types of bus master reads

When secure read protection is active, the affected flash blocks are readable by any bus master until a debugger is attached to the JTAG port and debug is enabled. If a debugger is attached and debug is enabled while an application is running, the next access to a flash read protected region terminates with a bus error.

Most instances of flash memory read protection in the SPC57EM80xx microcontroller are *not* configurable by the customer. An example of this is the 256-bit passwords stored in UTest flash memory. Password-based read protection *is* customer-configurable⁽¹⁾ and is similar to the password-based write protection mechanism detailed in [Section 37.3, Secure write protection](#).

There are two significant differences between password-based read protection and password-based write protection in the SPC57EM80xx microcontroller.

1. Read protection can only be applied to pre-defined groups of flash memory blocks. These groups, called "Read Lock groups", cannot be re-defined, meaning the blocks assigned to those groups are permanently assigned and cannot be assigned to other groups and no new groups can be created.
2. Read protection is only active either when a debugger is attached to the JTAG port and debug is enabled; or the lifecycle has been advanced to "Failure Analysis".

1. Secure read protection is user-configurable in that the default lock/unlock state for each mapped block at reset is controlled by user-created DCF records, giving the user the ability to determine the number of levels of password protection implemented. The read protection is applied to groups of blocks instead of individual blocks and the groups are not user-definable.

Another difference is that only the PASS_LOCK3_PGn register for each password group is used:

- PASS_LOCK3_PG0
- PASS_LOCK3_PG1
- PASS_LOCK3_PG2
- PASS_LOCK3_PG3

Note: The mappings of PASS_LOCK3_PGn register bits to flash blocks vary with different microcontrollers. See the flash memory section in the Device Configuration chapter of this reference manual for the mappings specific to the SPC57EM80xx microcontroller.

The next sections discuss implementing secure read protection and overriding secure write protection.

37.4.1 Implementing secure read protection

Implementation of secure read protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) are stored in a protected OTP area of device flash named UTest flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PGn–PASS_LOCK3_PGn) that have the same mapping of register bits to flash blocks as the flash module's SEL0–3 registers and LOCK0–3 registers. These are the same passwords used by secure write protection and other secure functions. The passwords are only created once.
2. DCF records must be created to define which flash memory blocks are to be locked. Those records determine the reset values of the PASS_LOCK3_PGn registers.
Like the other PASS_LOCKx_PGn registers, the PASS_LOCK3_PGn registers have two functions.
 - Reading the registers provides a lock status for each Read Locking group.
 - Changing the value of a mapped bit in a register changes the read protect status of the corresponding Read Locking group.

The layout of the PASS_LOCK3_PGn registers is shown in [Figure 439](#). The fields relevant to implementing secure read protection are RL4–RL0.

Figure 439. PASS_LOCK3_PGn Register

		0x010C LOCK3_PG0																			
		0x011C LOCK3_PG1																			
Offset		0x012C LOCK3_PG2																Access: Read/Write			
		0x013C LOCK3_PG3																			

Table 493. Sample low and mid address space flash block bit mapping (continued)

Start Address	End Address	Size (KB)	Partition	LOCK _n Register	LOCK _n Register Field Bit	LOCK _n DCF Record DATA Bit	Read Lock group	Block No.	Description
0x0061_0000	0x0061_FFFF	64	1	LOCK0	LOWLOCK[10]	26	3	10	Secure Code
0x0062_0000	0x0062_FFFF	64	1	LOCK0	LOWLOCK[11]	27	3	11	Secure Code
—	—	—	—	—	—	—	—	—	—
0x0068_0000	0x0068_3FFF	16	2	LOCK0	MIDLOCK[0]	0	4	12	Secure Data
0x0068_4000	0x0068_7FFF	16	3	LOCK0	MIDLOCK[1]	1	4	13	Secure Data
—	—	—	—	—	—	—	—	—	—
0x0040_0000	0x0040_3FFF	16	0	LOCK0	TSLOCK	31	0	32	UTEST

To provide two level of password protection for only Read Lock group 3 and leave all other groups unlocked, the RL3 bit can be set in any two of the four PASS_LOCK3_PG_n registers. For this example, we will choose password groups 0 and 1.

We need to create DCF records that define the reset values for all RL_n bits as 0, with the exception of PASS_LOCK3_PG0 and PASS_LOCK3_PG1. The information required is from the PASS section of [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#).

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0011	LOCK3_PG0
000_0000_0000_1000	000_0000_0100_0111	LOCK3_PG1
000_0000_0000_1000	000_0000_0100_1011	LOCK3_PG2
000_0000_0000_1000	000_0000_0100_1111	LOCK3_PG3

Referring to [Figure 436](#), the values for the DCF record are:

- For LOCK3_PG0
 - Data[31:0] is 0x0008_0000 (bits 0–31)
 - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
 - Address[16:2] is 0b000_0000_0100_0011 (bits 47–61)
 - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
 - Preferred Stop value is 0.
- For LOCK3_PG1
 - Data[31:0] is 0x0008_0000 (bits 0–31)
 - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
 - Address[16:2] is 0b000_0000_0100_0111 (bits 47–61)
 - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
 - Preferred Stop value is 0.
- For LOCK3_PG2
 - Data[31:0] is 0x0000_0000 (bits 0–31)
 - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
 - Address[16:2] is 0b000_0000_0100_1011 (bits 47–61)
 - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
 - Preferred Stop value is 0.
- For LOCK3_PG3
 - Data[31:0] is 0x0000_0000 (bits 0–31)
 - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
 - Address[16:2] is 0b000_0000_0100_1111 (bits 47–61)
 - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
 - Preferred Stop value is 0.

37.4.2 Overriding secure read protection

As noted previously, the PASS_LOCK3_PG n registers (see [Figure 439](#)) contain five read protection fields, RP0–RP4, that correspond to pre-defined Read Lock groups. To temporarily unlock a Read Lock group, the appropriate PASS_LOCK3_PG n [RL n] bits must be set to 0.

Like the register bits controlling secure write protection, the reset value of bits controlling secure read protection is determined by DCF record or, in the absence of a relevant DCF record, each bit resets to '1', causing the blocks associated with each read group to be read locked.

Example 11 Overriding secure read protection

Building on [Example 10](#), where Read Lock group 3 was protected by passwords 0 and 1, temporarily overriding secure read protection for that group is accomplished by first unlocking the PASS_LOCK3_PG0 and PASS_LOCK3_PG1 registers and then setting the RL3 bit to '0' in both.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN n) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS_CIN0.
3. Write a 0 to PASS_LOCK3_PG0[RL3] bit and verify a successful write.
4. Repeat the above steps for password 1.

The result of a successful sequence is that the blocks associated with the unlocked read lock group can be read by a debugger.

37.5 Debug port enable/disable

The mechanism for temporarily enabling the SPC57EM80xx debug port after it has been locked by its life cycle state is similar to the flash memory read protection mechanism detailed in [Section 37.4, Secure read protection](#). The difference is that the PASS_LOCK3_PG n [DBL] field is used instead of the RP0–RP4 fields.

37.6 Tamper detection

The Tamper Detection Module (TDM) provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

This section provides details on the mapping between DCF record bits and individual flash memory blocks to be selected for tamper detection.

See [Chapter 85: Tamper Detection Module \(TDM\)](#), for details on tamper detection, including the DCF clients and two additional memory-mapped registers.

There are two significant differences between selecting blocks for tamper detection and selecting blocks for other activities, e.g., a flash erase operation:

1. Selection of blocks for tamper detection protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. The selection of blocks for tamper detection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

The Tamper Detect section of DCF client list table ([Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#)) gives the CS and Address value combinations that correspond to groups of LOCK0–LOCK3 registers, similar to the flash module's LOCK0–LOCK3 registers.

The mapping of individual flash blocks to TDM DCF record data field bits is shown in included in the detailed flash block map in the flash section of the Device Configuration chapter.

37.6.1 Implementing tamper detection

Implementing tamper detection requires:

1. Creating the tamper detect diary
2. Assigning blocks to Tamper Detection Regions (TDRs)

The tasks are detailed in the following sections

37.6.2 Creating the tamper detect diary

Tamper detection in the SPC57EM80xx microcontroller is user-defined. The basic functionality provided is an erase counter, or diary, for 6 user-defined regions, called “tamper detect regions”, or TDRs, within the flash memory. Each TDR is defined using a set of DCF records that is similar in function and layout to the flash module's SEL n and LOCK n registers, which contain bits matched to specific flash blocks.

For a program or erase operation to be performed on a block included in a TDR, a record must be written to a “diary” that tracks flash modification operations. The format and size of records is customer-defined. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes.

Note: The limiting factor for tamper detection records is the diary size. The total diary size is given by the number of TDRs (6 for SPC57EM80xx) multiplied by 2 KB, the portion of the diary allotted to records associated with each TDR.

Since the minimum flash programming operation size is 8 bytes, the maximum number of tamper detect records for a given TDR is 256 8-byte entries.

The diary can be placed within any flash block in the flash array but to maintain the security of the TDR, there are two constraints:

- The block where the diary is placed must be assigned as OTP within the OTP registers (covered in the next section)
- The diary base address must be on a 4 KB boundary, therefore the least significant 12 bits of the base address must all be '0'

Warning: The tamper detect diary base address DCF record is a “write once” record. All subsequent tamper detect diary base address DCF records created are ignored.

Example 12 Creating the tamper detect diary

If the total space required for a diary is 12 KB (as in the case of 6 TDRs), any flash block can be used because the smallest flash blocks in the SPC57EM80xx microcontroller are 16 KB. Since the diary base address must be on a 4 KB boundary, though, if the diary base address is within a 16 KB flash block, it can only be assigned to the base address of the block or the address given by the base address plus 4 KB.

If the base address of the 16 KB block is 0x00FC4000, the base address of the diary is given by:

0b0000_0000_1111_1100_0000_0000_0000_0000 (16 KB base address)

or

0b0000_0000_1111_1100_0001_0000_0000_0000 (16 KB base address + 4 KB)

The base address of each TDR's section of the diary is:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

We need to create one DCF record that defines the base address for the diary. The information required is from the Tamper Detect section of [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#).

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0000	Diary Base Address

Referring to [Figure 436](#), the values for the DCF record are:

- For LOCK3_PG0
 - Data[31:0] is 0x00FC4000 (the base address)
 - CS[14:0] is 0b000_0000_0001_0000 (bits 32–46)
 - Address[16:2] is 0b000_0000_0000_0000 (bits 47–61)
 - Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
 - Preferred Stop value is 0.

See the Tamper Detect section of DCF client list table ([Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#)) for details.

Note: *User software must check whether the diary is full. If the diary area for a TDR is full, user software must program a DCF record to override blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry. This is detailed in [Section 37.6.4, Overriding tamper detection](#).*

37.6.3 Assigning blocks to Tamper Detection Regions (TDRs)

Tamper detection regions are defined by creating TDRx_LOCKn DCF records. For example, in a microcontroller with six tamper detection regions, the following records define the TDRs:

- TDR0_LOCK0, TDR0_LOCK1, TDR0_LOCK2, TDR0_LOCK3
- TDR1_LOCK0, TDR1_LOCK1, TDR1_LOCK2, TDR1_LOCK3
- TDR2_LOCK0, TDR2_LOCK1, TDR2_LOCK2, TDR2_LOCK3
- TDR3_LOCK0, TDR3_LOCK1, TDR3_LOCK2, TDR3_LOCK3
- TDR4_LOCK0, TDR4_LOCK1, TDR4_LOCK2, TDR4_LOCK3
- TDR5_LOCK0, TDR5_LOCK1, TDR5_LOCK2, TDR5_LOCK3

Each set of TDRxLOCKn DCF records mapped in exactly the same way as the flash module's SELn and LOCKn registers (see the detailed flash block map in the flash section of the Device Configuration chapter for the mapping).

In the TDRxLOCKn DCF records:

- A bit set to '1' means the corresponding flash block is assigned to this TDR
- A bit set to '0' means the corresponding flash block is NOT assigned to this TDR

The default state (no TDRs defined) is defined as 'Block NOT assigned to the TDR'.

37.6.4 Overriding tamper detection

When a TDR's diary section is full, a Tamper Override (TO) record must be created before a block assigned to the TDR can be programmed or erased.

The details of the data section of the DCF record used to override tamper detection is shown in [Figure 440](#).

Warning: Bits in this DCF record is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

Figure 440. Data portion of Tamper Region Override (TO) DCF record

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0

Example 13 Implementing secure write protection

Assume the diary Tamper Detection Region 3 is full

We need to create one DCF record that defines that overrides tamper detection for the tamper detection region whose diary is full. The information required is from the Tamper Detect section of [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#).

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0001	Tamper Region Override

Referring to [Figure 436](#), the values for the DCF record are:

- Data[31:0] is 0x0000_0008 (TOE3)
- CS[14:0] is 0b000_0000_0001_0000 (bits 32–46)
- Address[16:2] is 0b000_0000_0000_0001 (bits 47–61)
- Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
- Preferred Stop value is 0.

See the Tamper Detect section of DCF client list table ([Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#)) for details.

37.7 Implementing OTP

Any flash block within the flash array can be assigned, at any time, to be OTP. OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (i.e., flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block becomes OTP after the next reset.

Warning: After a flash block is configured as OTP, it cannot be changed back.

This capability resides within the Tamper Detection Module (TDM). The records are detailed in [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#). See the entries for the OTP_EN0, OTP_EN1, OTP_EN2, and OTP_EN3 DCF records in the Tamper Detect section of the table.

37.8 Implementing test mode disable

The SPC57EM80xx flash memory module includes a mechanism to disable manufacturer entry into test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method are able to have possible failures analyzed by manufacturer's failure analysts.

Test mode disable prevents all high voltage operations to the flash executed by the internal state-machine, as well as reads through the state machine, and reads through the Array Integrity state machine when using test mode interfaces.

Optionally, the customer can create a password to enable manufacturer entry into test mode.

37.8.1 Unconditional test mode disable seal

Warning: This process is not reversible. After completing these steps the SPC57EM80xx microcontroller is permanently prevented from entering test mode for factory analysis.

The steps for unconditionally disabling test mode are the same steps required for implementing passcode-protected entry into test mode with one exception—the passcode specified must be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To unconditionally disable test mode entry:

1. Select individual flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTest flash memory map.
2. Program one of the three *invalid* passcodes shown above into the Test Mode Override Passcode area shown in the UTest flash memory map.
3. Program the value 0x2D3C_4B5A into the Test Mode Disable Seal location specified in the UTest flash memory map.

After the next reset is asserted, Test Mode is permanently disabled.

Note: *Note: Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.*

The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

37.8.2 Passcode-protected test mode disable seal

The steps for implementing passcode-protected entry into test mode are the same required for unconditionally disabling test mode with the exception that —the passcode specified must NOT be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To implement passcode-protected test mode entry:

1. Select individual flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTest flash memory map.
2. Program a valid passcode into the Test Mode Override Passcode area shown in the UTest flash memory map.
3. Program the value 0x2D3C_4B5A into the Test Mode Disable Seal location specified in the UTest flash memory map.

After the next reset is asserted, Test Mode is permanently passcode protected.

Note: *Note: Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.*

The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

37.8.3 Selecting flash memory blocks for test mode disable seal

Selecting flash memory blocks for test mode disable seal protection is similar to selecting blocks for tamper detection in that it involves writing to non-register flash locations. In the case of the test mode disable seal function, there are two areas of UTest flash memory that have identical mapping to flash memory blocks. When the test mode disable seal takes effect, the mapped bits are logically ORed to determine the blocks to be protected.

Caution: The mapping of bits to blocks for this function is different from the mapping used for other functions. See the table in the device configuration chapter for the map.

Caution: Selecting a block for test mode disable requires a '0' in the mapped bit—not a '1' as with other functions.

The mapping of individual flash blocks to Test Mode Disable Block Select bits is shown in [Table 494](#).

Table 494. Mapping of Test Mode Disable Block Select bits to flash blocks

Block	Byte Offset ⁽¹⁾	Test Mode Disable Block Select Bit
32 KB Code Flash block1	0	0
32 KB Code Flash block0		1
16 KB HSM Code block5		2
16 KB Code Flash block4		3
16 KB Code Flash block3		4
16 KB Code Flash block2		5
16 KB Code Flash block1		6
16 KB BAF block0		7
—	1	8
—		9
—		10
—		11
64 KB HSM Code block3		12
64 KB HSM Code block2		13
64 KB Code Flash block1		14
64 KB Code Flash block0		15
—	2	16
—		17
—		18
—		19
—		20
—		21
16 KB HSM EEPROM block1		22
16 KB HSM EEPROM block0		23

Table 494. Mapping of Test Mode Disable Block Select bits to flash blocks (continued)

Block	Byte Offset ⁽¹⁾	Test Mode Disable Block Select Bit
—	3	24
—		25
—		26
—		27
—		28
—		29
—		30
—		31
—		32
—		33
—	4	34
—		35
64 KB EEPROM block3		36
64 KB EEPROM block2		37
64 KB EEPROM block1		38
64 KB EEPROM block0		39
—	5	40
—		41
—		42
—		43
—		44
—		45
—		46
—		47
—	6	48
256 KB Flash block7		49
256 KB Flash block6		50
256 KB Flash block5		51
256 KB Flash block4		52
256 KB Flash block3		53
256 KB Flash block2		54
256 KB Flash block1		55

Table 494. Mapping of Test Mode Disable Block Select bits to flash blocks (continued)

Block	Byte Offset ⁽¹⁾	Test Mode Disable Block Select Bit
256 KB Flash block0	7	56
—		57
—		58
256 KB Flash block13		59
256 KB Flash block12		60
256 KB Flash block11		61
256 KB Flash block10		62
256 KB Flash block9		63
—	8	64
—		65
—		66
—		67
—		68
—		69
—		70
—		71
—	9	72
—		73
—		74
—		75
—		76
—		77
—		78
—		79
—	10	80
—		81
—		82
—		83
—		84
—		85
—		86
—		87

Table 494. Mapping of Test Mode Disable Block Select bits to flash blocks (continued)

Block	Byte Offset ⁽¹⁾	Test Mode Disable Block Select Bit
—	11	88
—		89
—		90
—		91
—		92
—		93
—		94
—		95

1. The byte offset is from the start address of either the Test Mode Disable Block Select Group A or the start address of the Test Mode Disable Block Select Group B area of UTest flash. See [Table 16](#) in [Chapter 5: Memory Map](#), for the locations.

37.9 Security configuration planning

At minimum, designing the flash security framework consists of selecting the security features to be implemented and determining the scope of implementation for each selected feature.

In devices containing a hardware Security Module (HSM), significant planning must go into the implementation of HSM functions. The HSM is an independent software platform within the microcontroller. It is intended to be used to implement advanced security functions that are defined by customer software.

Note: Details of advanced security features, including the advanced security architecture and the Hardware Security Module (HSM) are contained in the SPC57EM80xx Microcontroller Security Reference Manual. Distribution of the document is restricted to qualified customers.

The basic security features available in the SPC57EM80xx microcontroller include:

- Secure write protection
- Secure read protection
- Debug port enable/disable
- One-time-programmable (OTP) flash memory block assignment
- Factory test mode disable

37.9.1 Hardware Security Module (HSM)

Details of the HSM are included in the *SPC57EM80xx Microcontroller Security Reference Manual*. It is mentioned here because the flash memory module includes blocks that can be configured for exclusive access by the HSM. If the HSM is to be used in an application it is advisable to protect HSM code by configuring it for exclusive access by the HSM. The blocks configurable for exclusive access by the HSM appear in [Figure 6](#) in [Chapter 3: Embedded memories](#). They are labeled as secure code flash or secure EEPROM emulation blocks. They are not configured for HSM-exclusive access by default, though—they are

available to act as secure flash areas but must be explicitly configured for that purpose. See the *SPC57EM80xx Microcontroller Security Reference Manual* for details.

37.9.2 Creating password groups

Most basic security features in the SPC57EM80xx microcontroller are password-based. Therefore, one of the early tasks that must be performed is the creation of four 256-bit passwords. They are programmed by the customer into the locations identified in the UTest memory map ([Table 16](#)) in [Chapter 5: Memory Map](#).

The UTest memory map fields to be written are:

- PASS Password Group 0
- PASS Password Group 1
- PASS Password Group 2
- PASS Password Group 3

Passwords must not be one of the following, which are considered insecure and therefore invalid:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

Warning: The password fields are in OTP UTest flash, so they can only be written once.

37.9.3 Planning secure write protection

Up to four levels of password-secured write protection can be assigned to any block. Note that by default, all blocks of user flash are write protected, so this could be more accurately viewed as planning which blocks of user flash will *not* have four levels of password-secured write protection. The write protection desired will depend on the usage of the flash blocks. Before creating DCF records to implement secure write protection, please thoroughly review the implementation steps detailed in [Section 37.3.1, Implementing secure write protection](#) and [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#).

37.9.4 Planning secure read protection

Up to four levels of password-secured write protection can be assigned to flash blocks in pre-defined Read Locking groups. Note that by default, all blocks of user flash are read protected, so this could be more accurately viewed as planning which blocks of user flash will *not* have four levels of password-secured read protection. The read protection desired will depend on the usage of the flash blocks. Before creating DCF records to implement secure read protection, please thoroughly review the implementation steps detailed in [Section 37.4.1, Implementing secure read protection](#) and [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#).

37.9.5 Planning debug port enable/disable

Please see [Section 37.5, Debug port enable/disable](#) and [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#).

37.9.6 Planning OTP flash memory block assignment

Any block of user flash can be configured as OTP. The need to configure blocks as OTP will depend on the data or code to be stored in the block. It is important to note that configuring a flash block as OTP is not reversible.

Before creating DCF records to implemented OTP flash blocks, please thoroughly review the implementation steps detailed in [Section 37.7, Implementing OTP](#) and [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#).

37.9.7 Planning factory test mode disable

Please see [Section 37.8, Implementing test mode disable](#) and [Table 112](#) in [Chapter 9: Device Configuration Format \(DCF\) Records](#).

38 Analog-to-Digital Converters (ADC) Configuration

38.1 ADC overview

This chapter is organized into the following sections:

- [Section 38.1, ADC overview](#)
 - [Section 38.1.1, ADC subsystem block diagram](#)
 - [Section 38.1.2, Analog input pin multiplexing](#)
- [Section 38.2, Configuration of ADC modules](#)
 - [Section 38.2.1, Sigma-Delta Analog-to-Digital Converter \(SDADC\)](#)
 - [Section 38.2.2, Successive Approximation Register Analog-to-Digital Converter \(SARADC\)](#)

SPC57EM80xx contains the following analog-to-digital converters (ADCs):

- 8 independent 12-bit Successive Approximation Register Analog-to-Digital Converters (SARADCs)
- 6 independent 16-bit Sigma-Delta Analog-to-Digital Converters (Sigma-Delta, or SDADCs)

These ADCs interface to external analog input pins:

Number of external analog input pins	Package
48	QFP176
48	216 FusionQuad ED
60	BGA292

Each ADC instance, whether SAR or Sigma-Delta, has independent analog input pin multiplexing, register interface, trigger and clock inputs, independent interrupt, and DMA capability. Each SAR and Sigma-Delta ADC has a dedicated peripheral bridge (PBRIDGE) interface for CPU access to the registers.

All SAR and Sigma-Delta ADCs share the same supply pins (VDD_HV_ADV/VSS_HV_ADV) and ground, but have separate reference pins. This is true for all packages.

Note that for compatibility reasons, the SARADC numbering omits SARADC_5. The SARADC_s on the device are referred to as SARADC_B (60:1 input mux), SARADC_0–SARADC_4, SARADC_6, and SARADC_7 (maximum 8:1 input mux for each).

38.1.1 ADC subsystem block diagram

[Figure 441](#) gives the block diagram for the high-level integration of the ADC sub-system for SPC57EM80xx. The primary intent of the diagram is to show that each ADC, whether SAR or Sigma-Delta, have independent register interfaces, interrupt/DMA requests, and trigger source selection.

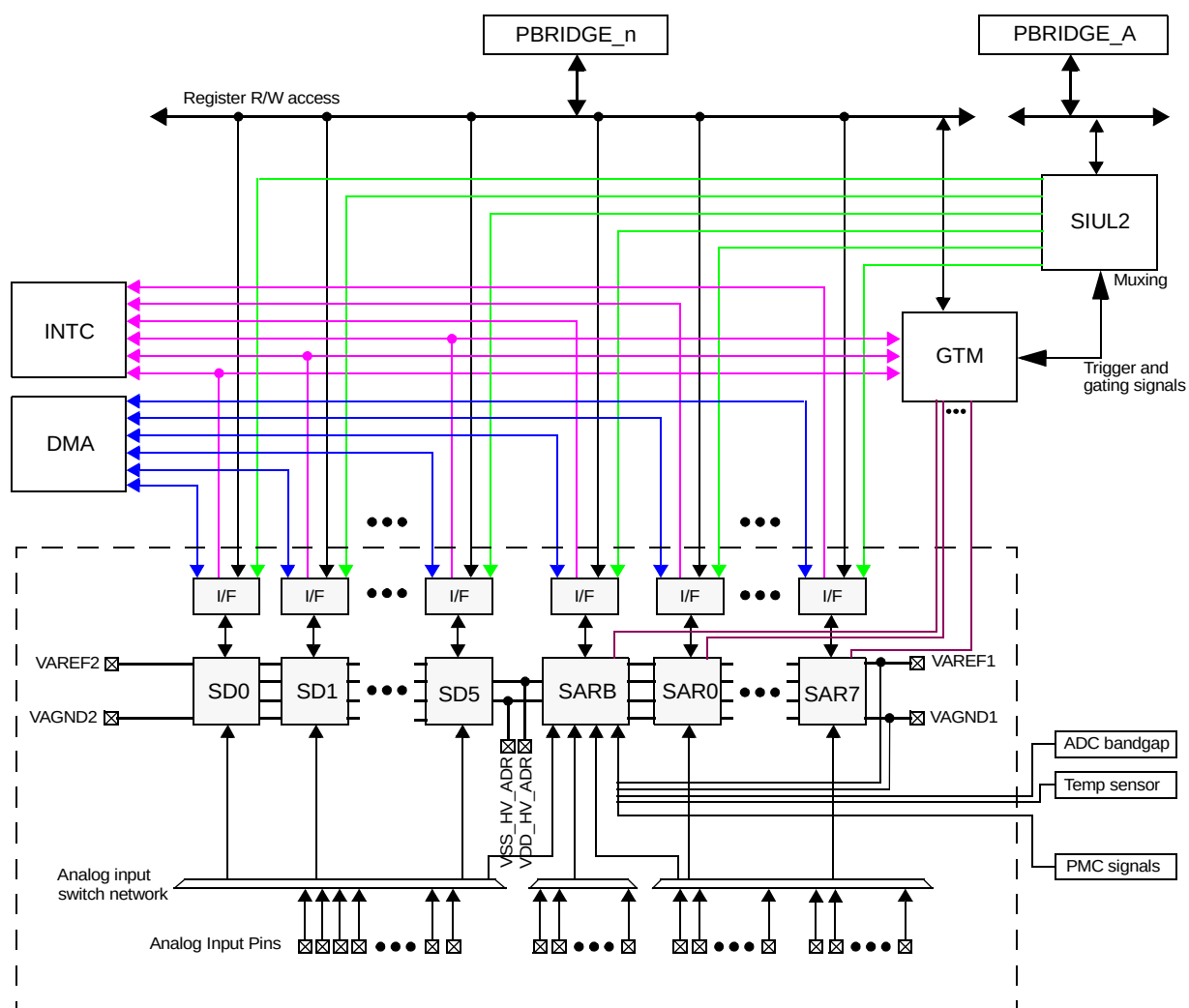


Figure 441. SPC57EM80xx ADC subsystem

38.1.2 Analog input pin multiplexing

There are 48 external analog input pins in the QFP176 package for SPC57EM80xx, and 60 in the BGA292 package. All analog input pins in either package are readable by SARADC_B. In addition to SARADC_B, most analog input pins are routed to either the Sigma-Delta or SARADC_s, but never both. [Table 495](#) gives the analog input pin assignment for both SPC57EM80xx QFP176 and 292 BGA packages.

Table 495. SPC57EM80xx analog input pin multiplexing

ADC instance	QFP176 (and 256 FusionQuad ED)		BGA292 (ED and PD)	
	Analog input pins	Mux input ⁽¹⁾	Analog input pins	Mux input ⁽¹⁾
SARADC_B	0:8, 10:13, 16:21, 24:29, 32:33, 35:39, 44:53, 56:61	0:8, 10:13, 16:21, 24:29, 32:33, 35:39, 44:53, 56:61	0:53, 56:61	0:53, 56:61
SARADC_0	AN4:AN7	4:7	AN4:AN7	4:7
SARADC_1	AN8, AN10:AN13	8, 10:13	AN8:AN15	8:15
SARADC_2	AN16:AN19	16:19	AN16:AN19, AN22, AN23	16:19, 22, 23
SARADC_3	AN26:AN29	26:29	AN26:AN31	26:31
SARADC_4	AN32:AN33, AN35	32:33, 35	AN32:AN35	32:35
SARADC_6	AN48:AN51	48:51	AN40:AN43, AN48:AN51	40:43, 48:51
SARADC_7	AN56:AN59	56:59	AN56:AN59	56:59
SDADC_0	AN2:AN3	AN[0]:AN[1]	AN2:AN3	AN[0]:AN[1]
SDADC_1	AN0:AN1	AN[0]:AN[1]	AN0:AN1	AN[0]:AN[1]
SDADC_2	AN20:AN21, AN24:AN25	AN[0]:AN[1], AN[2]:AN[3]	AN20:AN21, AN24:AN25	AN[0]:AN[1], AN[2]:AN[3]
SDADC_3	AN36:39, AN44:47	AN[0]:AN[7]	AN36:39, AN44:47	AN[0]:AN[7]
SDADC_4	AN52:AN53	AN[0]:AN[1]	AN52:AN53	AN[0]:AN[1]
SDADC_5	AN60:AN61	AN[0]:AN[1]	AN60:AN61	AN[0]:AN[1]

- For the Sigma-Delta mux inputs, even channel numbers (0,2,4,6) correspond to the positive terminal of the differential input, and odd channel numbers (1,3,5,7) correspond to the negative terminal.

38.2 Configuration of ADC modules

The following sections describe device-specific details for the following modules:

- [Section 38.2.1, Sigma-Delta Analog-to-Digital Converter \(SDADC\)](#)
- [Section 38.2.2, Successive Approximation Register Analog-to-Digital Converter \(SARADC\)](#)

38.2.1 Sigma-Delta Analog-to-Digital Converter (SDADC)

There are six independent Sigma-Delta ADCs on SPC57EM80xx. Refer to [Figure 441](#) for the ADC subsystem block diagram.

The following sections describe configuration details:

- [Section 38.2.1.1, Sigma-Delta ADC integration diagram](#)
- [Section 38.2.1.2, Sigma-Delta ADC analog input multiplexing](#)
- [Section 38.2.1.3, Sigma-Delta ADC clock sources](#)
- [Section 38.2.1.4, Sigma-Delta ADC starting/triggering](#)
- [Section 38.2.1.5, Sigma-Delta ADC external modulator](#)
- [Section 38.2.1.6, Conversion result interrupt/DMA](#)
- [Section 38.2.1.9, Sigma-Delta ADC analog input channel selection](#)

For detailed module information, refer to the Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface chapter.

38.2.1.1 Sigma-Delta ADC integration diagram

[Figure 442](#) shows the block diagram of the Sigma-Delta ADC integration on SPC57EM80xx. Note in the diagram that any unused Sigma-Delta ADC input can be read by SARADC_B. This is true for the unused input in single-ended operation.

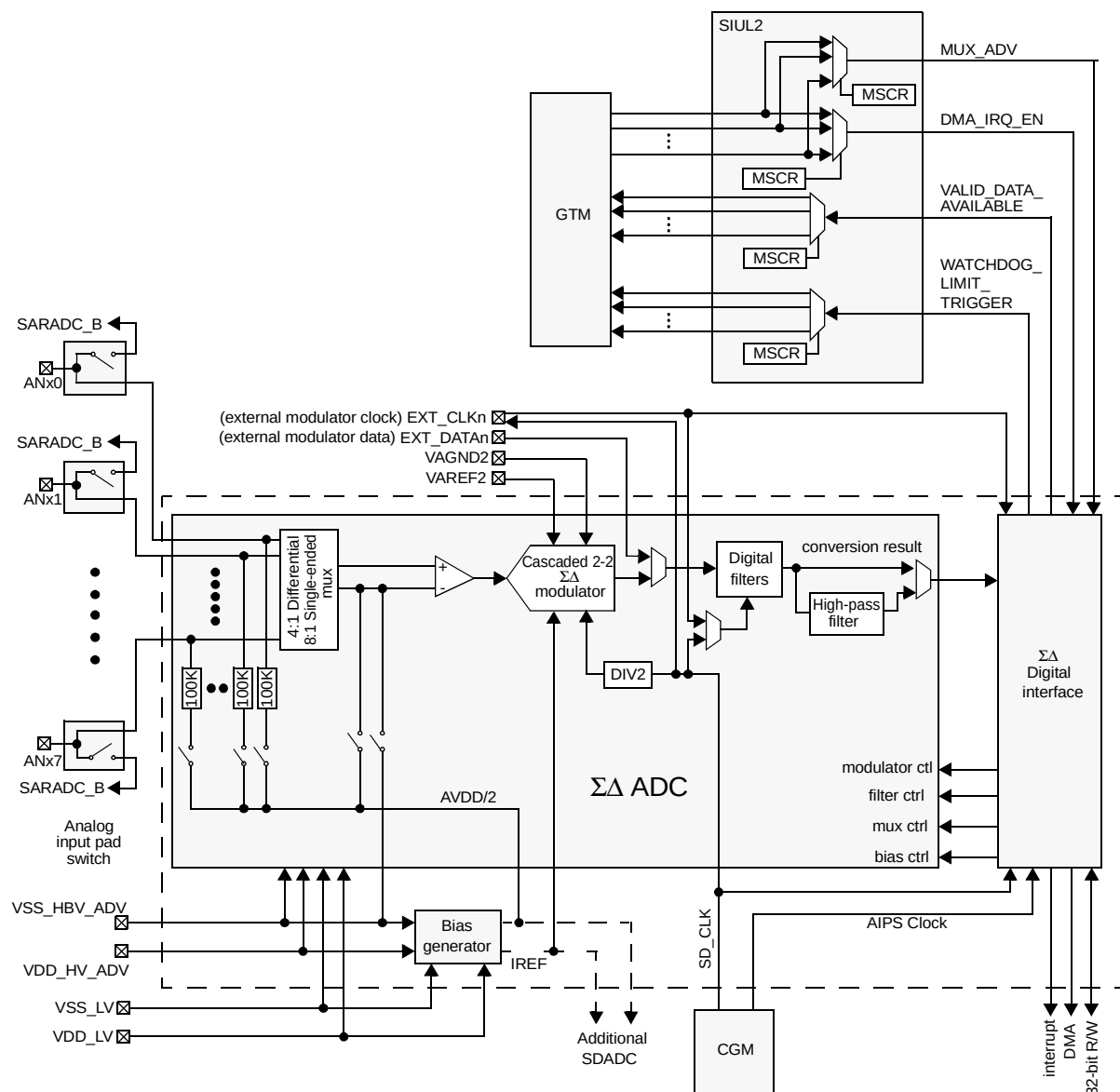


Figure 442. Sigma-Delta ADC integration diagram

38.2.1.2 Sigma-Delta ADC analog input multiplexing

The analog input multiplexing switches are contained within the Sigma-Delta ADCs. A single analog switch pad cell is used for the Sigma-Delta analog input pins in order to share the pin with SARADC_B. The connection to the Sigma-Delta ADC inputs is made to the pad pin through the analog switch pad cell, shown in [Figure 442](#), and is controlled as follows:

- SARADC_B analog inputs are switchable by the SIUL2_MSCRn[APC] bit.
- SDADC analog inputs have no SIUL2_MSCRn[APC] bit control, but are directly connected to the pad.

The mapping of analog input pins to the Sigma-Delta converters is given in [Table 495](#).

38.2.1.3 Sigma-Delta ADC clock sources

Each Sigma-Delta ADC digital interface on the SPC57EM80xx has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE_n_CLK. The module clock is sourced by the modulator clock (SD_CLK). The input clock for the Sigma-Delta modulator is the SD_CLK. [Figure 443](#) gives a block diagram of the required Sigma-Delta ADC clock architecture. See the clocking chapter, [Section 27.6.7, Sigma-Delta ADC clocking](#), for more detail on the clock sources.

The frequency range for the SD_CLK is 4–16 MHz. Both the analog and digital blocks of the ADC contain an internal clock mux for external modulator mode. The external modulator clock (EXT_CLK) can be input to the device, or generated from SD_CLK from the CGM. The pad input/output settings in the corresponding SIUL2 MSCR pad control registers determine the external modulator clock source to the ADC blocks. The EXT_CLK pins are multiplexed on multiple external pins on the device. Selection of the EXT_CLK pin to use for the external modulator clock for each ADC is done in the SIUL2 MSCR input multiplexing registers. Each Sigma-Delta ADC has three external pins to select from for an external modulator clock.

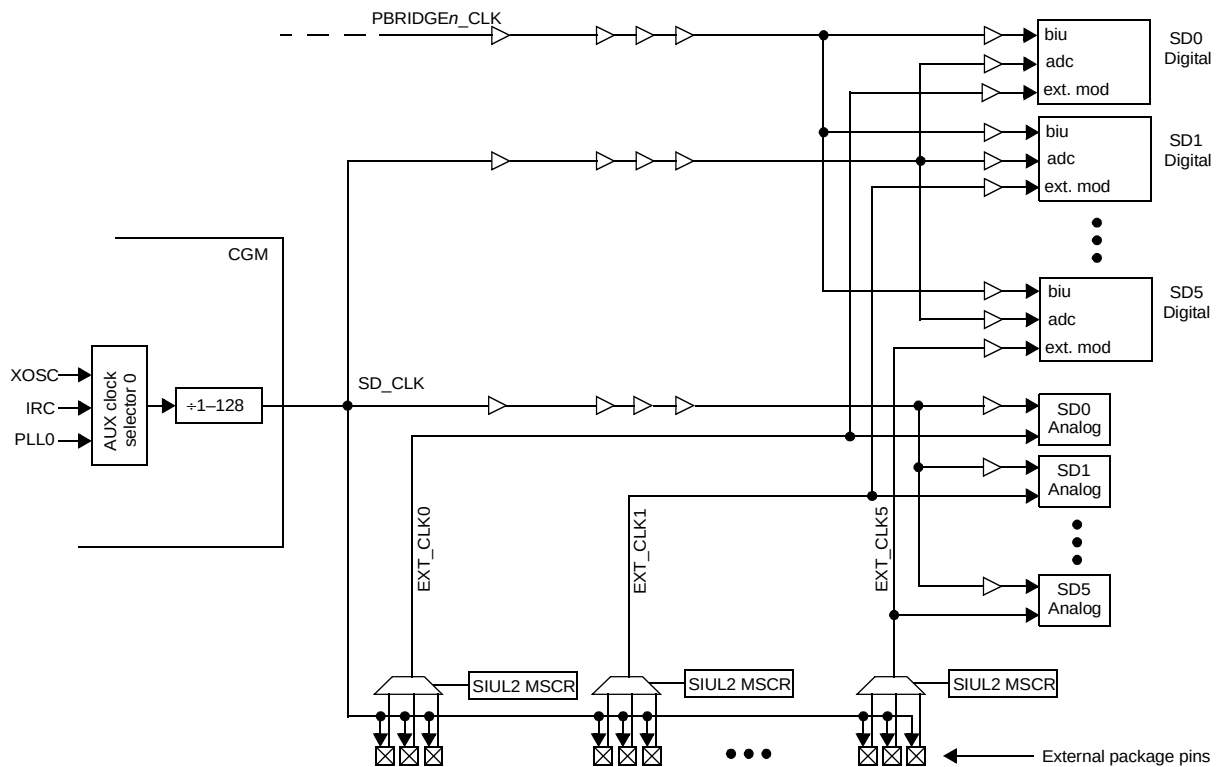


Figure 443. Sigma-Delta ADC clock diagram

38.2.1.3.1 Clock Skew

To avoid crosstalk issues, all Sigma-Delta ADCs run at the same clock frequency. Independent programmable oversampling rates are provided to support the varying bandwidths of interest on each ADC in the application. Therefore, all Sigma-Delta analog blocks are clocked from the output of a single divider of the peripheral clock at the chip level.

38.2.1.4 Sigma-Delta ADC starting/triggering

Each Sigma-Delta ADC on SPC57EM80xx supports starting the converter with a software write to an enable bit in a register in the digital interface.

38.2.1.4.1 Simultaneous start of multiple Sigma-Delta converters

A method to start multiple Sigma-Delta converters simultaneously by a single software write to a control register is available on the device. Simultaneously in this case means that all selected converters start sampling their inputs on the same SD_CLK clock edge.

The digital interface of the Sigma-Delta ADC contains the control register for simultaneous start of the ADCs. When the register is written to start multiple converters, an output from the digital interface is asserted. The hardware trigger input of each Sigma-Delta ADC is connected to this simultaneous start output. This way the user controls which ADCs are selected by enabling the hardware trigger of desired ADCs before writing to the starting simultaneous converters.

The connections for the simultaneous start of the converters is given in [Figure 444](#).

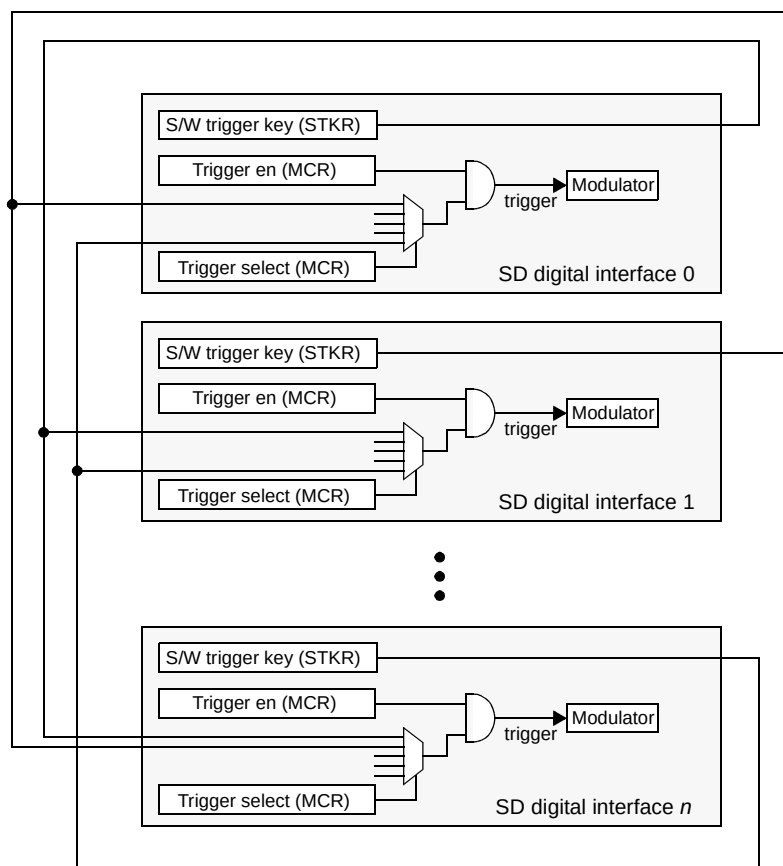


Figure 444. Sigma-Delta ADC simultaneous start connectivity

38.2.1.4.2 Sigma-Delta mux input advance and wraparound

The mux wraparound feature is enabled and controlled in the digital interface of the Sigma-Delta ADC. The advancing of the mux can be controlled in the interface by software or by external GTM channels. The mapping of GTM output channels to the mux advance input for each Sigma-Delta ADC for each device is given in the SIUL2 MSCR registers.

38.2.1.5 Sigma-Delta ADC external modulator

38.2.1.5.1 External modulators

The Sigma-Delta ADC supports an interface to external Sigma-Delta modulators. In this mode, the Sigma-Delta ADC internal modulator is disabled, and a serial bit stream from an external modulator is clocked into the back-end digital filters of the on-chip Sigma-Delta ADC. Currently, the AMC1203 external modulator is supported by the Sigma-Delta ADC. It is possible that other external 2nd order modulators with clock rates less than 10 MHz and a compatible electrical interface (see SPC57EM80xx Data Sheet) could be supported.

The external modulator interface for each Sigma-Delta ADC on SPC57EM80xx is shown in [Figure 445](#).

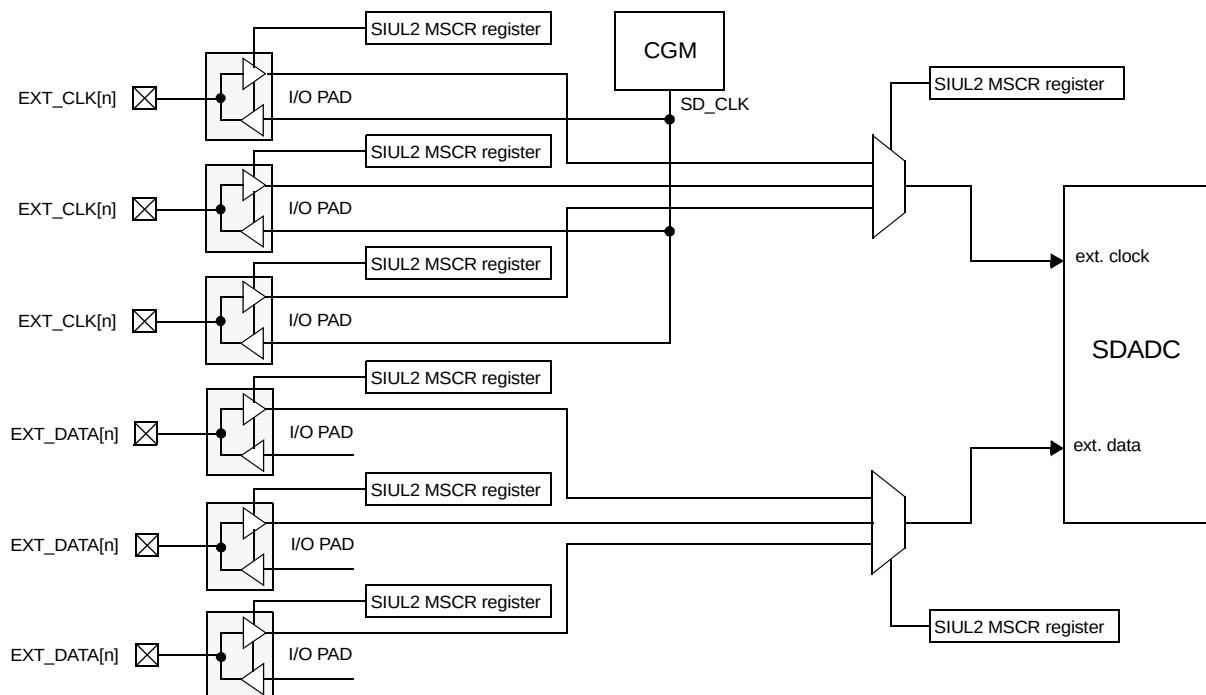


Figure 445. Sigma-Delta ADC external modulator interface

38.2.1.5.2 External modulator clock/data inputs

Two external pins are required for each Sigma-Delta ADC to support an external modulator, an input data pin EXT_DATA (bitstream data), and an input/output clock pin EXT_CLK (bitstream clock). When the external modulator is used, the clock and data inputs to the Sigma-Delta ADC digital filters come from the EXT_CLK and EXT_DATA pins, instead of the internal modulator as shown in [Figure 453](#). External modulator operation is enabled in the ADC digital interface registers. The internal modulator is disabled. The EXT_CLK and

EXT_DATA inputs are multiplexed on port input pins. For each Sigma-Delta ADC, there are three sets of EXT_CLK/EXT_DATA pins, and selection of those pins for each ADC is done in the SIUL2 MSCR registers.

38.2.1.5.3 External modulator clock output

SPC57EM80xx provides an option for an output clock from the device to the external modulator. When providing an output clock, the SD_CLK from the CGM is connected to the EXT_CLK output pin. Both the input and output buffer of the EXT_CLK pin must be enabled in the corresponding SIUL2 MSCR register, in order for both the external modulator and internal filters to be clocked from the SD_CLK. This is shown in [Figure 445](#).

38.2.1.6 Conversion result interrupt/DMA

A DMA or interrupt can be generated for each conversion for each ADC. The Sigma-Delta ADC has an 8-deep result FIFO.

The Sigma-Delta ADC digital interface has an input signal that is used to disable interrupt and DMA requests for conversion results. The input signal is connected to outputs from the GTM timer block. The digital interface has a register bit for enabling and disabling the interrupt/DMA gating feature.

When the signal from the GTM is asserted, no interrupt or DMA request is generated for any conversion results. The polarity of the gating signal is such that when the signal is high, interrupt and DMA requests are passed. When the gating signal is low, interrupt and DMA requests are blocked.

The gating signal for each ADC is driven by several GTM channels. See [Chapter 16: System Integration Unit Lite2 \(SIUL2\)](#) for the specific mapping of GTM channels to Sigma-Delta interrupt/DMA gating signals.

38.2.1.7 Conversion Limit Watchdog Output to GTM

Each Sigma-Delta ADC provides an optional conversion limit check for upper and lower bounds. The enable bit and the two 16-bit limit registers for the feature are included in the SDADC wrapper. If a conversion result exceeds one of the limits, a flag bit is asserted in a status register in the SDADC wrapper. The flag bit is output to the GTM for each SDADC. The mapping of conversion limit watchdog outputs to GTM channels is defined in the SIUL2 MSCR registers.

38.2.1.8 SDADC_MCR[TRIGSEL] definitions

For this chip, the following input triggers apply:

TRIGSEL	Trigger Input Selection This field selects which input will be used for hardware-triggered conversions. 0000 SDADC_0 0001 SDADC_1 0010 SDADC_2 0011 SDADC_3 0100 SDADC_4 0101 SDADC_5 0110 – 1110 Reserved 1111 MUX_ADV <i>Note: Selecting the input trigger source to be from the output trigger of the same SDADC is not prohibited by hardware, but is not supported, and will result in the SDADC never responding to the input trigger.</i>
---------	---

38.2.1.9 Sigma-Delta ADC analog input channel selection

For the Sigma-Delta mux inputs AN[x], the positive and negative input terminals are selected in the Sigma-Delta ADC digital interface Channel Selection Register (CSR).

This section shows analog channel selections for each combination of Module Configuration Register (MCR) field values MCR_MODE and MCR_VCOMSEL. The following table provides a quick reference key to the MCR fields used in the SD ADC analog input selection tables that follow it.

Table 496. Register fields for SD ADC analog input channel selection

Field	Description
MCR_VCOMSEL	Common Voltage Bias Selection This bit selects the common voltage bias for the negative input terminal of SDADC during single-ended mode (MODE=1). 0 Negative input terminal is biased with VREFN. 1 Negative input terminal is biased with VREFP/2 (half scale bias).
MCR_MODE	Mode selection 0 Differential input mode is selected. 1 Single-ended input mode is selected.
CSR_ANCHSEL	Analog Channel Selection Based on single-ended or differential mode of operation (MODE bit of MCR), common mode voltage selection (VCOMSEL bit of MCR), this bit-field defines the connectivity of analog inputs to either positive or negative polarity terminals of Sigma-Delta ADC, as shown in the following tables: – Table 497 for SDADC_0 – Table 498 for SDADC_1 – Table 499 for SDADC_2 – Table 500 for SDADC_3 – Table 501 for SDADC_4 – Table 502 for SDADC_5

Table 497. SDADC_0 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 498. SDADC_1 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 499. SDADC_2 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 500. SDADC_3 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	AN[4]	AN[5]
		011	AN[6]	AN[7]
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 501. SDADC_4 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 502. SDADC_5 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

38.2.2 Successive Approximation Register Analog-to-Digital Converter (SARADC)

There are eight independent SARADC_s on SPC57EM80xx. Refer to [Figure 441](#) for the ADC sub-system block diagram.

The following sections describe configuration details:

- [Section 38.2.2.1, SAR ADC integration diagram](#)
- [Section 38.2.2.2, SAR ADC analog input pin multiplexing](#)
- [Section 38.2.2.3, SAR ADC clock sources](#)
- [Section 38.2.2.4, SAR ADC triggering](#)
- [Section 38.2.2.5, SAR ADC alternate reference](#)
- [Section 38.2.2.6, SAR absolute voltage reference](#)
- [Section 38.2.2.7, SAR ADC diagnostic](#)
- [Section 38.2.2.8, SARADC channel assignment](#)
- [Section 38.2.2.9, SAR ADC register definitions](#)

For detailed module information, refer to the Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

38.2.2.1 SAR ADC integration diagram

[Figure 446](#) shows the block diagram of the integration of the eight SAR ADCs on SPC57EM80xx.

SARADC_B implements a bias generator converting the VDD_HV_ADR_S, 1/3 VDD_HV_ADR_S, 2/3 VDD_HV_ADR_S, and VSS_HV_ADR_S reference points through the internal 20 kΩ source impedance. See the SARADC block diagram in the Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

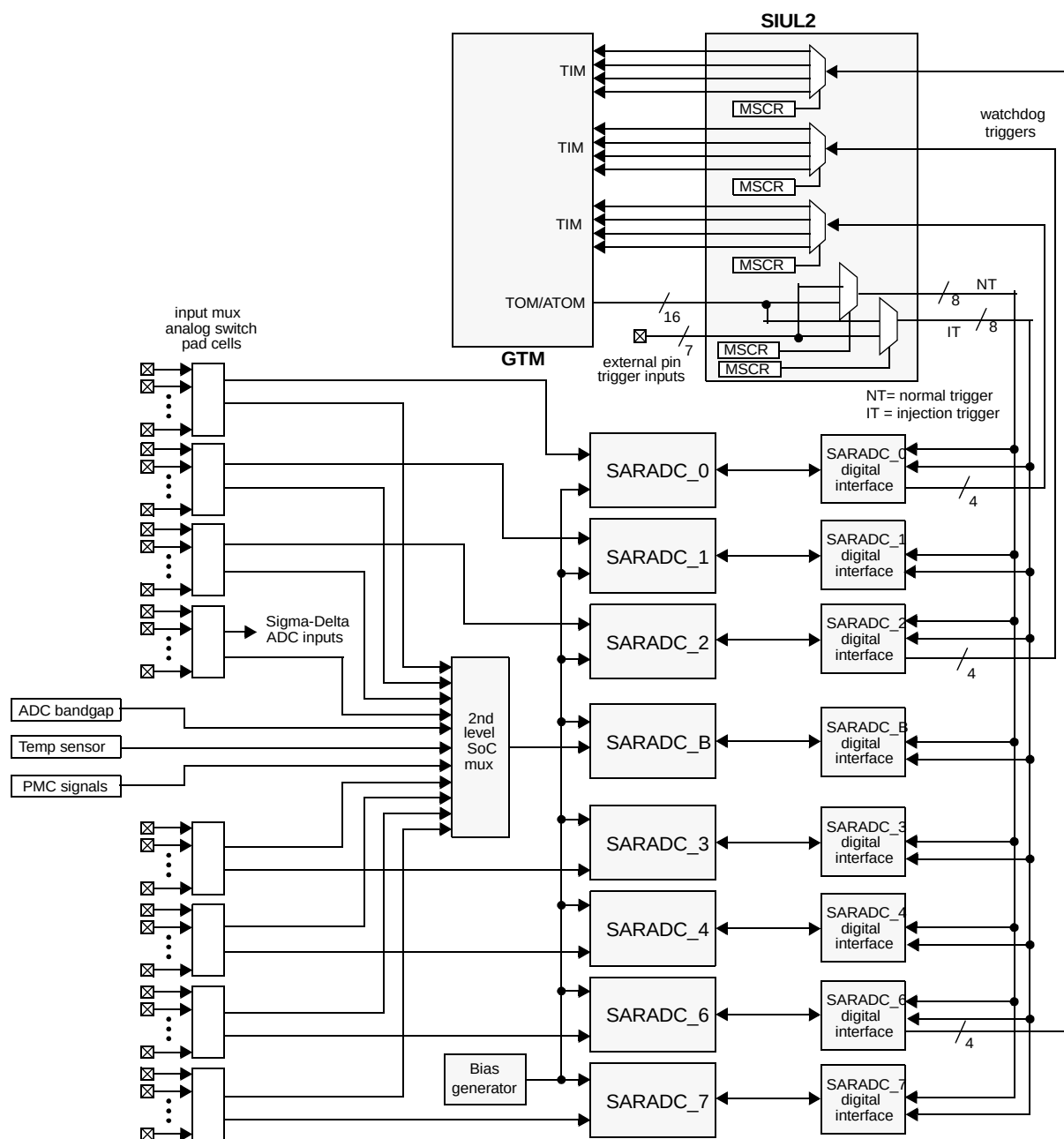


Figure 446. SAR ADC integration diagram

38.2.2.2 SAR ADC analog input pin multiplexing

All analog input pins routed to SARADC_B and another fast SAR ADC are multiplexed with a dual analog input switch pad cell. Analog input pins that only connect to SARADC_B use a single analog switch pad cell.

Note: Simultaneous sampling of two ADCs on same analog input is not allowed.

A second level of multiplexing is required for SARADC_B to reduce the parasitic capacitance on the analog input due to the 60:1 input mux. The second level muxing is done in an SoC level IP block so that it can be optimally placed for routing. The analog switches in the second level mux block are controlled by the SAR ADC digital interface block.

The mapping of analog input pins to the SAR converters is given in [Table 495](#).

38.2.2.3 SAR ADC clock sources

Each SAR ADC digital interface on SPC57EM80xx has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE $_n$ _CLK. The module clock is sourced by the ADC clock (SAR_CLK). See [Section 27.2, Clock generation](#) for more detail on the clock sources.

The input clock for the SAR ADC analog blocks is clocked by the SAR_CLK. [Figure 447](#) shows a block diagram of the SAR ADC clock architecture. The maximum frequency for the clock input to the converter is 14.6 MHz.

To avoid crosstalk issues, all SAR ADCs run at the same clock frequency. The SAR digital interface is clocked from the PBRIDGE $_n$ _CLK clock. All SAR analog blocks are clocked from the output of a single divider of the SAR_CLK at the device level.

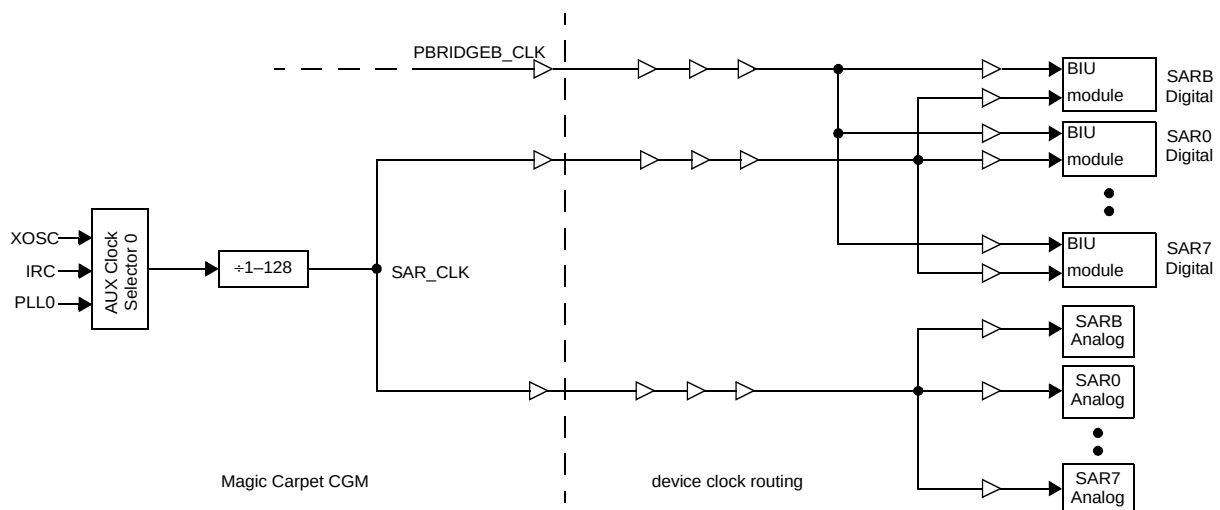


Figure 447. SAR ADC clock diagram

38.2.2.4 SAR ADC triggering

Each SAR ADC has 16 independent trigger inputs for starting conversion on an analog input. Selection from the 16 trigger inputs is done inside the SAR ADC digital wrapper. Each SAR ADC on the device has an independent set of trigger inputs, but the same sources are used for each SAR ADC, so only one trigger input is required for each SAR ADC.

The trigger sources for each SAR ADC on the device are either external input pins or GTM channels. Each external input can selectively trigger any of the SAR ADCs on the device. The synchronization to the PBRIDGE clock and digital filtering of the external input pin triggers is done in the system integration logic.

Each trigger input supports software selectable options for rising edge, falling edge, rising and falling edge, or level-based trigger assertion. The trigger selection for each SAR ADC and edge/level selection is located in the SIUL2. The SPC57EM80xx SoC diagram for the SAR ADC trigger inputs is given in [Figure 448](#).

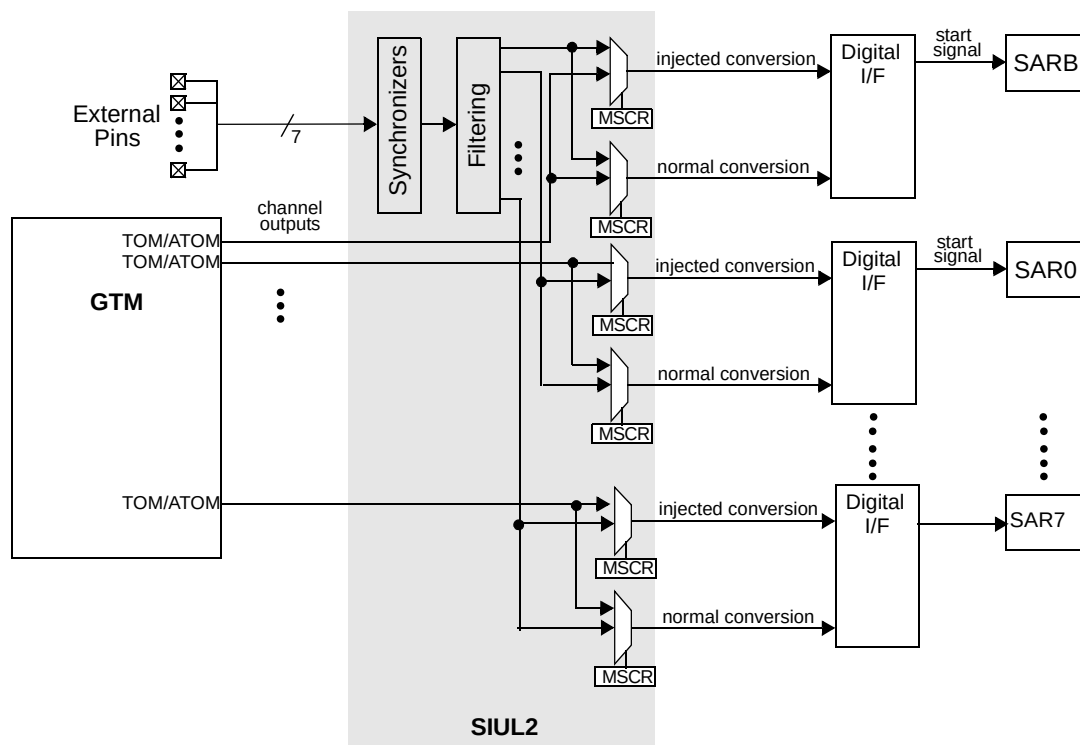


Figure 448. SARADC input trigger diagram

Trigger source selection for each SAR ADC is given in the SIUL2 chapter of this document.

38.2.2.5 SAR ADC alternate reference

The SAR ADCs support connection to more than one conversion reference. The alternate references are muxed with analog input pins on SPC57EM80xx. The analog input pins used as alternate references to the respective SAR ADCs are given in [Table 503](#).

Table 503. SARADC alternate references

SARADC	Alternate reference pin
SARADC_B	AN[8]
SARADC_2	AN[16]

To meet safety requirements, the reference for specific SAR analog inputs cannot be changed. For SPC57EM80xx, these are SARADC_B channels 4–7 and 35, and all other SAR analog inputs. For these SAR ADC instances that do not enable an alternate reference, ICDRn[REFSEL] is a read-only bit. There is no option to connect these pins as analog references for the device for safety considerations.

Alternate reference selection is enabled only for the following channels, for which ICDRn[REFSEL] is read-write bit:

- SARADC_B channels (0–3, 8–34, 36–95): See [Table 4, Section 38.2.2.9, SAR ADC register definitions](#).
- All SAR2 channels (16–19, 22–23): See [Table 9, Section 38.2.2.9, SAR ADC register definitions](#).

38.2.2.6 SAR absolute voltage reference

SPC57EM80xx provides an internal, absolute voltage reference that can be read by SARADC_B. The ADC conversion channel number for the absolute reference, ADC Bandgap Reference, is given in [Table 505](#).

38.2.2.7 SAR ADC diagnostic

38.2.2.7.1 Self Test features

Following are the Self Test features:

- The ADCBIAS provides four different voltage levels defined as GND, $V_{ref}/3$, $2*V_{ref}/3$ and V_{ref} . The self test internal reference voltage precision is provided in the SARn ADC electrical specification table of the device datasheet.
- The impedance requirement has been satisfied putting a physical poly resistor $R_{out} = 20\text{ k}\Omega$ typical characterized by a process and temperature variation of $\pm 30\%$.
- The impedance of the switches that are connecting the different references to the ADC input has not been considered in the impedance calculation.
- The impedance of the switches from the reference generation to the ADC input has been matched with the one of the switches that are connecting the device pin to the ADC input.
- The sampling time of SARADC_B needs to be extended to 2.2 ms due to long settling time induced by the high impedance of the source (20 k Ω).
- The SAR ADC Digital Interface provides the necessary control to select the required switches S1–S7 (shown in [Figure 449 \(Self Test implementation\)](#)) to perform the Self Test.

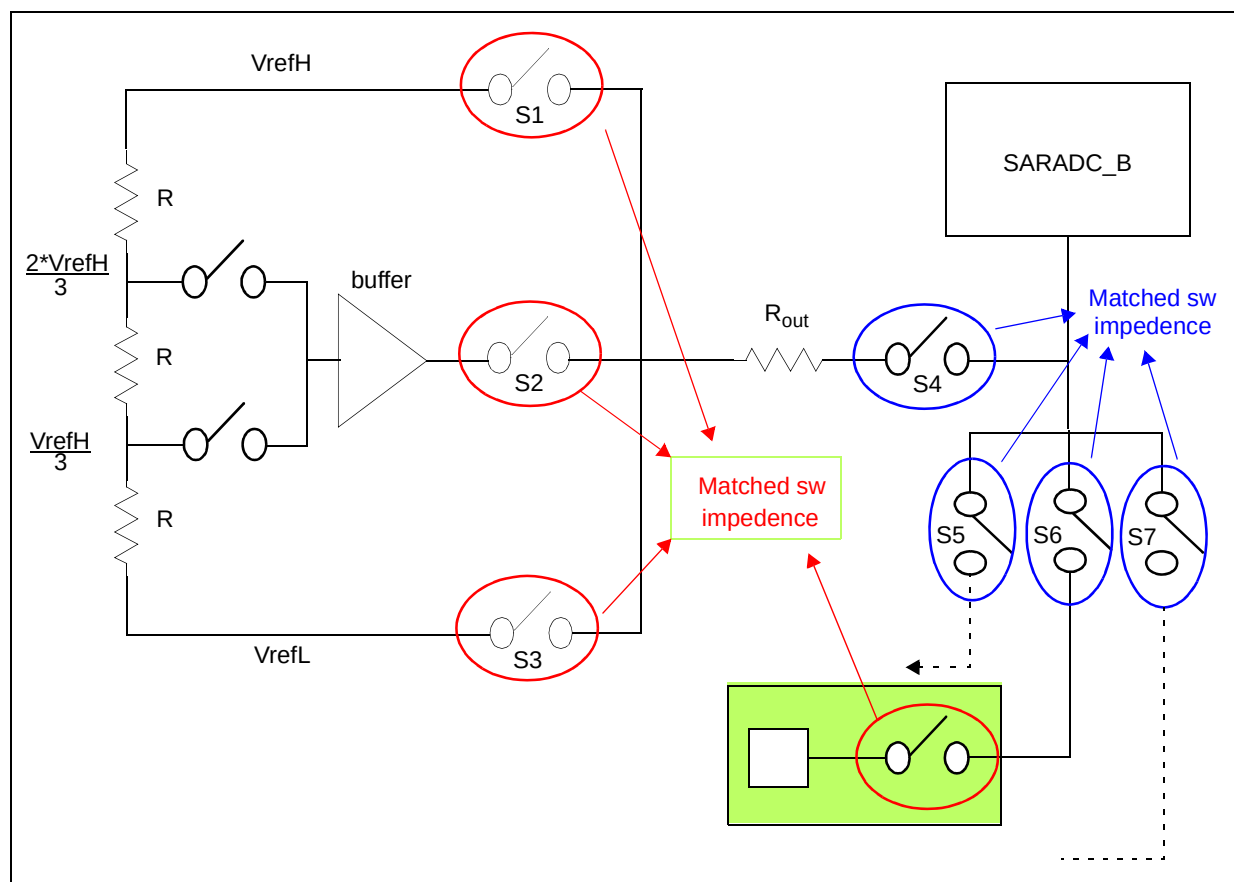


Figure 449. Self Test implementation

38.2.2.7.2 Internal reference

The SARB ADC provides the ability to sample and convert four internal voltages through a 20 kΩ source impedance. These voltages are as follows:

- $V_{DD_HV_ADR_S}$
- $\frac{1}{3} V_{DD_HV_ADR_S}$
- $\frac{2}{3} V_{DD_HV_ADR_S}$
- $V_{SS_HV_ADR_S}$

Conversion of an internal reference voltage works the same as that for a normal conversion, with the exception of the conversion time. Due to the 20 kΩ source impedance, the sampling time is much higher than the normal time.

Enabling the sampling and selection of an internal reference is independent of input channel selection for SARADC_B.

38.2.2.7.3 Analog input pin programmable pullup/pulldown

An open or short circuit condition is detectable within the device for analog input pins. There is a programmable pullup/pulldown pad cell at the input pad. This allows for application testing of the mux input logic. The analog input pin diagram with the pullup/pulldown is shown in [Figure 450](#). The pullup/pulldown is included inside the analog input pad cell. The pullup/pulldown is also included in the single switch analog input pad cell.

Control of the pullup/pulldown is independently done in the SIUL2 MSCR registers for each analog input pin.

Only the analog input pins specified in [Table 495](#) have the pullup/pulldown feature.

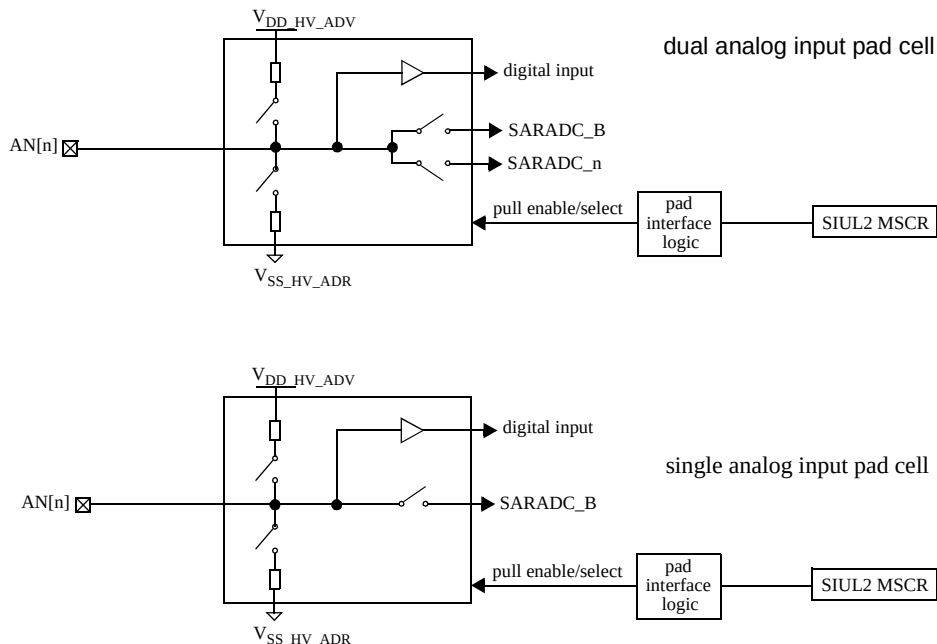


Figure 450. Analog input pin pullup/pulldown pad cell

38.2.2.7.4 Analog input pin switch to $V_{SS_HV_ADV}$

Two analog input pins on SPC57EM80xx have a programmable switch to $V_{SS_HV_ADV}$ to discharge the input. This switch is implemented on analog input pins AN7 and AN35 for SPC57EM80xx. The analog input diagram for AN35 is given in [Figure 451](#). The default state of the switch for both inputs is OFF.



Figure 451. Analog input AN35 — discharge switch to AVSS

The ground switches for AN7/AN35 are controlled independently from the input switch to the ADC. The ground switch for AN7/AN35 is controlled in input multiplexing MSCR registers (MSCR[WPDE]) in the SIUL2. The MSCR register provides selection between open or closed switch, with open being the default state.

The ground switches can be used to periodically discharge the associated channels.

38.2.2.8 SARADC channel assignment

The input multiplexer channel assignments for the SARADCs is provided in [Table 504](#).

38.2.2.8.1 Internal voltage monitor channel assignments

Each internal voltage monitor has a unique channel assignment, and is treated by software like any other A/D conversion. Access to the internal monitor channels is required for SARADC_B only. The internal voltage monitor channel assignments for SARADC_B are given in [Table 505](#).

Table 504. SAR analog input channel assignment

Analog input pin	SARADC_B input channel	Fast SAR	Fast SAR channel	Description
AN[3:0]	3–0	—	—	Analog input pin
AN[7:4]	7–4	SARADC_0	7–4	Analog input pin, fixed reference, AN[7] gnd switch
AN[15:8]	15–8	SARADC_1	15–8	Analog input pin
AN[19:16]	19–16	SARADC_2	19–16	Analog input pin
AN[21:20]	21–20	—	—	Analog input pin
AN[23:22]	23–22	SARADC_2	23–22	Analog input pin
AN[25:24]	25–24	—	—	Analog input pin
AN[29:26]	29–26	SARADC_3	29–26	Analog input pin
AN[31:30]	31–30	SARADC_3	31–30	Analog input pin
AN[35:32]	35–32	SARADC_4	35–32	Analog input pin, AN[35] fixed reference and gnd switch
AN[39:36]	39–36	—	—	Analog input pin
AN[43:40]	43–40	SARADC_6	43–40	Analog input pin
AN[47:44]	47–44	—	—	Analog input pin
AN[51:48]	51–48	SARADC_6	51–48	Analog input pin
AN[53:52]	53–52	—	—	Analog input pin
AN[59:56]	59–56	SARADC_7	59–56	Analog input pin
AN[61:60]	61–60	—	—	Analog input pin
—	95–62	—	—	Reserved

Table 505. SARADC_B analog test channel assignment

SARADC_B input channel	Fast SAR	Fast SAR channel	Description
96	—	—	VDD_HV_PMC
97	—	—	VDD_HV_ADV

Table 505. SARADC_B analog test channel assignment (continued)

SARADC_B input channel	Fast SAR	Fast SAR channel	Description
98	—	—	VDD_HV_JTAG_OSC
99	—	—	VDD_HV_FLA
100	—	—	VDD_HV_IO_MAIN
101	—	—	VDD_HV_IO_FLEX
102	—	—	Reserved
103	—	—	Reserved
104	—	—	VDD_LV
105	—	—	Reserved
106	—	—	Crude PMC bandgap reference
107	—	—	Reserved
108	—	—	Trimmed PMC bandgap reference
109	—	—	VSS_HV_ADV (Ground Supply for ADC)
110	—	—	PMC generic input 0
111	—	—	Reserved
112	—	—	HV regulator supply LVD falling edge 280_C
113	—	—	HV regulator supply HVD rising edge 600_C
114	—	—	HV flash supply LVD falling edge 280_F
115	—	—	HV flash supply HVD rising edge 360_F
116	—	—	LV core supply LVD falling edge 114_C
117	—	—	LV core supply HVD falling edge 140_C
118	—	—	LV flash supply LVD falling edge 114_F
119	—	—	Reserved
120	—	—	Temperature sensor
121	—	—	ADC bandgap reference
122	—	—	Reserved
123	—	—	Reserved
124	—	—	SAR BIAS 0 — VSS_HV_ADR_S through 20 K Ω source impedance
125	—	—	SAR BIAS 1 — 1/3 (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 K Ω source impedance
126	—	—	SAR BIAS 2 — 2/3 (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 K Ω source impedance
127	—	—	SAR BIAS 3 — (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 K Ω source impedance

38.2.2.8.2 SARADC_B interface to PMC

The connection of the PMC to SARADC_B ADC is shown in [Figure 452](#). SARADC_B ADC channel 110 connects to the PMC. The PMC contains an analog mux that passes internal voltage nodes in addition to those given in [Table 505](#). The PMC mux is programmed in a register in the PMC digital interface. To convert an internal PMC voltage, the user programs the PMC analog mux selection, and then programs SARADC_B to convert PMC channel 110. See the Power Management chapter of this document for more info on additional internal voltages that can be read by the SARADC_B ADC.

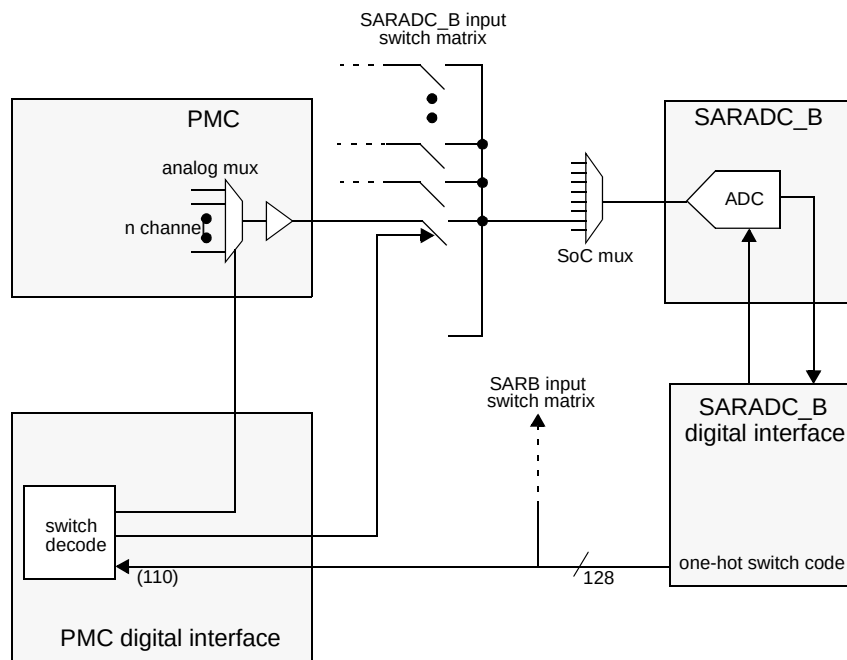


Figure 452. SARADC_B ADC read of internal PMC voltages

38.2.2.9 SAR ADC register definitions

Each of the following sections shows register definitions for one SAR ADC instance:

- [Section 38.2.2.9.1, SARADC_B register definitions](#)
- [Section 38.2.2.9.2, SARADC_0 register definitions](#)
- [Section 38.2.2.9.3, SARADC_1 register definitions](#)
- [Section 38.2.2.9.4, SARADC_2 register definitions](#)
- [Section 38.2.2.9.5, SARADC_3 register definitions](#)
- [Section 38.2.2.9.6, SARADC_4 register definitions](#)
- [Section 38.2.2.9.7, SARADC_6 register definitions](#)
- [Section 38.2.2.9.8, SARADC_7 register definitions](#)

The following tables provide a key for the register definition tables. Test channels and external channels are only used for SARADC_B.

Table 506. SAR ADC register descriptions

Name	Description	User Access
MCR	Main Configuration Register	read/write
MSR	Main Status Register	read
ISR	Interrupt Status Register	read
ICIPR n	Internal Channel Interrupt Pending Registers	read
IMR	Interrupt Mask Register	read/write
ICIMR n	Internal Channel Interrupt Mask Registers	read/write
WTISR	Watchdog Threshold Interrupt Status Register	read
WTIMR	Watchdog Threshold Interrupt Mask Register	read/write
DMAE	DMA Enable Register	read/write
ICDSR n	Internal Channel DMA Select Registers	read/write
WTHRHLR0–3	Watchdog Threshold Registers 0–3	read/write
CTR n	Conversion Timing Registers 0–3	read/write
ICNCMR n	Internal Channel Normal Conversion Mask Registers	read/write
ICJCMR n	Internal Channel Injected Conversion Mask Registers	read/write
PEDR	Power Down Exit Delay Register	read/write
ICDR n	Internal Channel Data Registers 0–95	mixed
WTHRHLR4–15	Watchdog Threshold Registers 4–15	read/write
ICWSELR n	Internal Channel Watchdog Select Registers	read/write
ICWENR n	Internal Channel Watchdog Enable Registers	read/write
ICAWORR n	Internal Channel Analog Watchdog Out of Range Registers	read

Table 507. SAR ADC test channel register descriptions (SARB only)

Name	Description	User Access
TCIPR	Test Channel Interrupt Pending Register	read
TCIMR	Test Channel Interrupt Mask Register	read/write
TCDSR	Test Channel DMA Select Register	read/write
TCNCMR	Test Channel Normal Conversion Mask Register	read/write
TCJCMR	Test Channel Injected Conversion Mask Register	read/write
TCWSELR n	Test Channel Watchdog Select Register	read/write
TCWENR	Test Channel Watchdog Enable Register	read/write
TCAWORR	Test Channel Analog Watchdog Out of Range Register	read
TCCAPR n	Test Channel Connection with Analog Pin Registers	read/write
TCDR n	Test Channel Data Register 96-127	read/write

Table 508. SAR ADC external channel register descriptions (SARB only)

Name	Description	User Access
ECDSDR	External Channel Decode Signals Delay Register	read/write
ECIPR n	External Channel Interrupt Pending Registers	read
ECIMR n	External Channel Interrupt Mask Registers	read/write
ECDSR n	External Channel DMA Select Registers	read/write
ECNCMR n	External Channel Normal Conversion Mask Registers	read/write
ECJCMR n	External Channel Injected Conversion Mask Registers	read/write
ECWSELR n	External Channel Watchdog Select Registers	read/write
ECWENR n	External Channel Watchdog Enable Registers	read/write
ECAWORR n	External Channel Analog Watchdog Out of Range registers	read
ECMICR n	External Channel Mapping to Internal Channel Registers	read/write
ECDR128-191	External Channel Data Registers 128-191	read/write

38.2.2.9.1 SARADC_B register definitions

Table 509. SARADC_B register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_MCR	R				0										0	0	0
	W	OWREN	WLSIDE	MODE		NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN		JEDGESEL	JTRGSEQ			
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADCB_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADCB_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c

Table 509. SARADC_B register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ICIPR0	R	EOC_CH[31]	EOC_CH[30]	EOC_CH[29]	EOC_CH[28]	EOC_CH[27]	EOC_CH[26]	EOC_CH[25]	EOC_CH[24]	EOC_CH[23]	EOC_CH[22]	EOC_CH[21]	EOC_CH[20]	EOC_CH[19]	EOC_CH[18]	EOC_CH[17]	EOC_CH[16]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOC_CH[15]	EOC_CH[14]	EOC_CH[13]	EOC_CH[12]	EOC_CH[11]	EOC_CH[10]	EOC_CH[9]	EOC_CH[8]	EOC_CH[7]	EOC_CH[6]	EOC_CH[5]	EOC_CH[4]	EOC_CH[3]	EOC_CH[2]	EOC_CH[1]	EOC_CH[0]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ICIPR1	R	0	0	EOC_CH[61]	EOC_CH[60]	EOC_CH[59]	EOC_CH[58]	EOC_CH[57]	EOC_CH[56]	0	0	EOC_CH[53]	EOC_CH[52]	EOC_CH[51]	EOC_CH[50]	EOC_CH[49]	EOC_CH[48]
	W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
	R	EOC_CH[47]	EOC_CH[46]	EOC_CH[45]	EOC_CH[44]	EOC_CH[43]	EOC_CH[42]	EOC_CH[41]	EOC_CH[40]	EOC_CH[39]	EOC_CH[38]	EOC_CH[37]	EOC_CH[36]	EOC_CH[35]	EOC_CH[34]	EOC_CH[33]	EOC_CH[32]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADCB_ICIMR0	R	IM_CH[31]	IM_CH[30]	IM_CH[29]	IM_CH[28]	IM_CH[27]	IM_CH[26]	IM_CH[25]	IM_CH[24]	IM_CH[23]	IM_CH[22]	IM_CH[21]	IM_CH[20]	IM_CH[19]	IM_CH[18]	IM_CH[17]	IM_CH[16]
	W	IM_CH[15]	IM_CH[14]	IM_CH[13]	IM_CH[12]	IM_CH[11]	IM_CH[10]	IM_CH[9]	IM_CH[8]	IM_CH[7]	IM_CH[6]	IM_CH[5]	IM_CH[4]	IM_CH[3]	IM_CH[2]	IM_CH[1]	IM_CH[0]
	R	IM_CH[15]	IM_CH[14]	IM_CH[13]	IM_CH[12]	IM_CH[11]	IM_CH[10]	IM_CH[9]	IM_CH[8]	IM_CH[7]	IM_CH[6]	IM_CH[5]	IM_CH[4]	IM_CH[3]	IM_CH[2]	IM_CH[1]	IM_CH[0]
	W	IM_CH[15]	IM_CH[14]	IM_CH[13]	IM_CH[12]	IM_CH[11]	IM_CH[10]	IM_CH[9]	IM_CH[8]	IM_CH[7]	IM_CH[6]	IM_CH[5]	IM_CH[4]	IM_CH[3]	IM_CH[2]	IM_CH[1]	IM_CH[0]
SARADCB_ICIMR1	R	0	0	IM_CH[61]	IM_CH[60]	IM_CH[59]	IM_CH[58]	IM_CH[57]	IM_CH[56]	0	0	IM_CH[53]	IM_CH[52]	IM_CH[51]	IM_CH[50]	IM_CH[49]	IM_CH[48]
	W			IM_CH[45]	IM_CH[44]	IM_CH[43]	IM_CH[42]	IM_CH[41]	IM_CH[40]			IM_CH[37]	IM_CH[36]	IM_CH[35]	IM_CH[34]	IM_CH[33]	IM_CH[32]
	R	IM_CH[47]	IM_CH[46]	IM_CH[45]	IM_CH[44]	IM_CH[43]	IM_CH[42]	IM_CH[41]	IM_CH[40]	IM_CH[39]	IM_CH[38]	IM_CH[37]	IM_CH[36]	IM_CH[35]	IM_CH[34]	IM_CH[33]	IM_CH[32]
	W	IM_CH[47]	IM_CH[46]	IM_CH[45]	IM_CH[44]	IM_CH[43]	IM_CH[42]	IM_CH[41]	IM_CH[40]	IM_CH[39]	IM_CH[38]	IM_CH[37]	IM_CH[36]	IM_CH[35]	IM_CH[34]	IM_CH[33]	IM_CH[32]

Table 509. SARADC_B register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC_B_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC_B_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC_B_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC_B_ICDSR0	R	DS_CH[31]	DS_CH[30]	DS_CH[29]	DS_CH[28]	DS_CH[27]	DS_CH[26]	DS_CH[25]	DS_CH[24]	DS_CH[23]	DS_CH[22]	DS_CH[21]	DS_CH[20]	DS_CH[19]	DS_CH[18]	DS_CH[17]	DS_CH[16]
	W																
	R	DS_CH[15]	DS_CH[14]	DS_CH[13]	DS_CH[12]	DS_CH[11]	DS_CH[10]	DS_CH[9]	DS_CH[8]	DS_CH[7]	DS_CH[6]	DS_CH[5]	DS_CH[4]	DS_CH[3]	DS_CH[2]	DS_CH[1]	DS_CH[0]
	W																
SARADC_B_ICDSR1	R	0	0	DS_CH[61]	DS_CH[60]	DS_CH[59]	DS_CH[58]	DS_CH[57]	DS_CH[56]	0	0	DS_CH[53]	DS_CH[52]	DS_CH[51]	DS_CH[50]	DS_CH[49]	DS_CH[48]
	W																
	R	DS_CH[47]	DS_CH[46]	DS_CH[45]	DS_CH[44]	DS_CH[43]	DS_CH[42]	DS_CH[41]	DS_CH[40]	DS_CH[39]	DS_CH[38]	DS_CH[37]	DS_CH[36]	DS_CH[35]	DS_CH[34]	DS_CH[33]	DS_CH[32]
	W																
SARADC_B_WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC_B_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																

Table 509. SARADC_B register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ICNCMR0	R																
	W																
	R																
	W																
SARADCB_ICNCMR1	R																
	W																
	R																
	W																
SARADCB_ICJCMR0	R																
	W																
	R																
	W																
SARADCB_ICJCMR1	R																
	W																
	R																
	W																
SARADCB_ICDR0-3, ICDR8-34, ICDR36-95	R																
	W																
	R																
	W																
SARADCB_PEDR	R																
	W																
	R																
	W																

Table 509. SARADC_B register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ICDR4-7, ICDR35	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADCB_ WTHRHLR4– WTHRHLR7	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADCB_ICWSELR0	R	0	WSEL_CH7			0	WSEL_CH6			0	WSEL_CH5			0	WSEL_CH4		
	W																
	R	0	WSEL_CH3			0	WSEL_CH2			0	WSEL_CH1			0	WSEL_CH0		
	W																
SARADCB_ICWSELR1	R	0	WSEL_CH15			0	WSEL_CH14			0	WSEL_CH13			0	WSEL_CH12		
	W																
	R	0	WSEL_CH11			0	WSEL_CH10			0	WSEL_CH9			0	WSEL_CH8		
	W																
SARADCB_ICWSELR2	R	0	WSEL_CH23			0	WSEL_CH22			0	WSEL_CH21			0	WSEL_CH20		
	W																
	R	0	WSEL_CH19			0	WSEL_CH18			0	WSEL_CH17			0	WSEL_CH16		
	W																
SARADCB_ICWSELR3	R	0	WSEL_CH31			0	WSEL_CH30			0	WSEL_CH29			0	WSEL_CH28		
	W																
	R	0	WSEL_CH27			0	WSEL_CH26			0	WSEL_CH25			0	WSEL_CH24		
	W																
SARADCB_ICWSELR4	R	0	WSEL_CH39			0	WSEL_CH38			0	WSEL_CH37			0	WSEL_CH36		
	W																
	R	0	WSEL_CH35			0	WSEL_CH34			0	WSEL_CH33			0	WSEL_CH32		
	W																
SARADCB_ICWSELR5	R	0	WSEL_CH47			0	WSEL_CH46			0	WSEL_CH45			0	WSEL_CH44		
	W																
	R	0	WSEL_CH43			0	WSEL_CH42			0	WSEL_CH41			0	WSEL_CH40		
	W																

Table 509. SARADC_B register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ICWSELR6	R	0	0	0	0	0	0	0	0	0	WSEL_CH53			0	WSEL_CH52		
	W																
	R	0	WSEL_CH51			0	WSEL_CH50			0	WSEL_CH49			0	WSEL_CH48		
	W																
SARADCB_ICWSELR7	R	0	0	0	0	0	0	0	0	0	WSEL_CH61			0	WSEL_CH60		
	W																
	R	0	WSEL_CH59			0	WSEL_CH58			0	WSEL_CH57			0	WSEL_CH56		
	W																
SARADCB_ICWENR0	R	WEN_CH[31]		WEN_CH[30]		WEN_CH[29]		WEN_CH[28]		WEN_CH[27]		WEN_CH[26]		WEN_CH[25]		WEN_CH[24]	
	W																
	R	WEN_CH[15]		WEN_CH[14]		WEN_CH[13]		WEN_CH[12]		WEN_CH[11]		WEN_CH[10]		WEN_CH[9]		WEN_CH[8]	
	W	WEN_CH[15]		WEN_CH[14]		WEN_CH[13]		WEN_CH[12]		WEN_CH[11]		WEN_CH[10]		WEN_CH[9]		WEN_CH[8]	
SARADCB_ICWENR1	R	0	0	WEN_CH[61]		WEN_CH[60]		WEN_CH[59]		WEN_CH[58]		WEN_CH[57]		WEN_CH[56]		WEN_CH[55]	
	W																
	R	WEN_CH[47]		WEN_CH[46]		WEN_CH[45]		WEN_CH[44]		WEN_CH[43]		WEN_CH[42]		WEN_CH[41]		WEN_CH[40]	
	W	WEN_CH[47]		WEN_CH[46]		WEN_CH[45]		WEN_CH[44]		WEN_CH[43]		WEN_CH[42]		WEN_CH[41]		WEN_CH[40]	
SARADCB_ICAWORR0	R	AWOR_CH[31]		AWOR_CH[30]		AWOR_CH[29]		AWOR_CH[28]		AWOR_CH[27]		AWOR_CH[26]		AWOR_CH[25]		AWOR_CH[24]	
		AWOR_CH[31]		AWOR_CH[30]		AWOR_CH[29]		AWOR_CH[28]		AWOR_CH[27]		AWOR_CH[26]		AWOR_CH[25]		AWOR_CH[24]	
	W	w1c		w1c		w1c		w1c		w1c		w1c		w1c		w1c	
		w1c		w1c		w1c		w1c		w1c		w1c		w1c		w1c	
	R	AWOR_CH[15]		AWOR_CH[14]		AWOR_CH[13]		AWOR_CH[12]		AWOR_CH[11]		AWOR_CH[10]		AWOR_CH[9]		AWOR_CH[8]	
		AWOR_CH[15]		AWOR_CH[14]		AWOR_CH[13]		AWOR_CH[12]		AWOR_CH[11]		AWOR_CH[10]		AWOR_CH[9]		AWOR_CH[8]	
	W	w1c		w1c		w1c		w1c		w1c		w1c		w1c		w1c	
		w1c		w1c		w1c		w1c		w1c		w1c		w1c		w1c	

Table 509. SARADC_B register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ICAWORR1	R	0	0	AWOR_CH[61]	AWOR_CH[60]	AWOR_CH[59]	AWOR_CH[58]	AWOR_CH[57]	AWOR_CH[56]	0	0	AWOR_CH[53]	AWOR_CH[52]	AWOR_CH[51]	AWOR_CH[50]	AWOR_CH[49]	AWOR_CH[48]
	W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
	R	AWOR_CH[47]	AWOR_CH[46]	AWOR_CH[45]	AWOR_CH[44]	AWOR_CH[43]	AWOR_CH[42]	AWOR_CH[41]	AWOR_CH[40]	AWOR_CH[39]	AWOR_CH[38]	AWOR_CH[37]	AWOR_CH[36]	AWOR_CH[35]	AWOR_CH[34]	AWOR_CH[33]	AWOR_CH[32]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 510. SARADC_B test channel register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_TCIPR	R	EOC_CH[127]	EOC_CH[126]	EOC_CH[125]	EOC_CH[124]	0	EOC_CH[122]	EOC_CH[121]	EOC_CH[120]	EOC_CH[119]	EOC_CH[118]	EOC_CH[117]	EOC_CH[116]	EOC_CH[115]	EOC_CH[114]	EOC_CH[113]	EOC_CH[112]
	W	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	0	EOC_CH[110]	0	EOC_CH[108]	0	EOC_CH[106]	0	EOC_CH[104]	0	0	EOC_CH[101]	EOC_CH[100]	EOC_CH[99]	EOC_CH[98]	EOC_CH[97]	EOC_CH[96]
	W		w1c		w1c		w1c		w1c			w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_TCIMR	R	IM_CH[127]	IM_CH[126]	IM_CH[125]	IM_CH[124]	0	IM_CH[122]	IM_CH[121]	IM_CH[120]	IM_CH[119]	IM_CH[118]	IM_CH[117]	IM_CH[116]	IM_CH[115]	IM_CH[114]	IM_CH[113]	IM_CH[112]
	W	IM_CH[127]	IM_CH[126]	IM_CH[125]	IM_CH[124]		IM_CH[122]	IM_CH[121]	IM_CH[120]	IM_CH[119]	IM_CH[118]	IM_CH[117]	IM_CH[116]	IM_CH[115]	IM_CH[114]	IM_CH[113]	IM_CH[112]
	R	0	IM_CH[110]	0	IM_CH[108]	0	IM_CH[106]	0	IM_CH[104]	0	0	IM_CH[101]	IM_CH[100]	IM_CH[99]	IM_CH[98]	IM_CH[97]	IM_CH[96]
	W		IM_CH[110]		IM_CH[108]		IM_CH[106]		IM_CH[104]			IM_CH[101]	IM_CH[100]	IM_CH[99]	IM_CH[98]	IM_CH[97]	IM_CH[96]

Table 510. SARADC_B test channel register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_TCDSR	R	DS_CH[127]	DS_CH[126]	DS_CH[125]	DS_CH[124]	0	DS_CH[122]	DS_CH[121]	DS_CH[120]	DS_CH[119]	DS_CH[118]	DS_CH[117]	DS_CH[116]	DS_CH[115]	DS_CH[114]	DS_CH[113]	DS_CH[112]
	W																
	R	0	DS_CH[110]	0	DS_CH[108]	0	DS_CH[106]	0	DS_CH[104]	0	0	DS_CH[101]	DS_CH[100]	DS_CH[99]	DS_CH[98]	DS_CH[97]	DS_CH[96]
	W		DS_CH[110]		DS_CH[108]		DS_CH[106]		DS_CH[104]			DS_CH[101]	DS_CH[100]	DS_CH[99]	DS_CH[98]	DS_CH[97]	DS_CH[96]
SARADCB_TCNCMR	R	NCE_CH[127]	NCE_CH[126]	NCE_CH[125]	NCE_CH[124]	0	NCE_CH[122]	NCE_CH[121]	NCE_CH[120]	NCE_CH[119]	NCE_CH[118]	NCE_CH[117]	NCE_CH[116]	NCE_CH[115]	NCE_CH[114]	NCE_CH[113]	NCE_CH[112]
	W																
	R	0	NCE_CH[110]	0	NCE_CH[108]	0	NCE_CH[106]	0	NCE_CH[104]	0	0	NCE_CH[101]	NCE_CH[100]	NCE_CH[99]	NCE_CH[98]	NCE_CH[97]	NCE_CH[96]
	W		NCE_CH[110]		NCE_CH[108]		NCE_CH[106]		NCE_CH[104]			NCE_CH[101]	NCE_CH[100]	NCE_CH[99]	NCE_CH[98]	NCE_CH[97]	NCE_CH[96]
SARADCB_TCJCMR	R	JCE_CH[127]	JCE_CH[126]	JCE_CH[125]	JCE_CH[124]	0	JCE_CH[122]	JCE_CH[121]	JCE_CH[120]	JCE_CH[119]	JCE_CH[118]	JCE_CH[117]	JCE_CH[116]	JCE_CH[115]	JCE_CH[114]	JCE_CH[113]	JCE_CH[112]
	W																
	R	0	JCE_CH[110]	0	JCE_CH[108]	0	JCE_CH[106]	0	JCE_CH[104]	0	0	JCE_CH[101]	JCE_CH[100]	JCE_CH[99]	JCE_CH[98]	JCE_CH[97]	JCE_CH[96]
	W		JCE_CH[110]		JCE_CH[108]		JCE_CH[106]		JCE_CH[104]			JCE_CH[101]	JCE_CH[100]	JCE_CH[99]	JCE_CH[98]	JCE_CH[97]	JCE_CH[96]
SARADCB_TCWSELR0	R	0	0	0	0	0	0	0	0	0	WSEL_CH101			0	WSEL_CH100		
	W																
	R	0	WSEL_CH99			0	WSEL_CH98			0	WSEL_CH97			0	WSEL_CH96		
	W																
SARADCB_TCWSELR1	R	0	0	0	0	0	WSEL_CH110			0	0	0	0	0	WSEL_CH108		
	W																
	R	0	0	0	0	0	WSEL_CH106			0	0	0	0	0	WSEL_CH104		
	W																
SARADCB_TCWSELR2	R	0	WSEL_CH119			0	WSEL_CH118			0	WSEL_CH117			0	WSEL_CH116		
	W																
	R	0	WSEL_CH115			0	WSEL_CH114			0	WSEL_CH113			0	WSEL_CH112		
	W																

Table 510. SARADC_B test channel register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_TCWSELR3	R	0	WSEL_CH127			0	WSEL_CH126			0	WSEL_CH125			0	WSEL_CH124		
	W																
	R	0	0	0	0	0	WSEL_CH122			0	WSEL_CH121			0	WSEL_CH120		
	W																
SARADCB_TCWENR	R	WEN_CH[127]	WEN_CH[126]	WEN_CH[125]	WEN_CH[124]	0	WEN_CH[122]	WEN_CH[121]	WEN_CH[120]	WEN_CH[119]	WEN_CH[118]	WEN_CH[117]	WEN_CH[116]	WEN_CH[115]	WEN_CH[114]	WEN_CH[113]	WEN_CH[112]
	W																
	R	0	WEN_CH[110]	0	WEN_CH[108]	0	WEN_CH[106]	0	WEN_CH[104]	0	0	WEN_CH[101]	WEN_CH[100]	WEN_CH[99]	WEN_CH[98]	WEN_CH[97]	WEN_CH[96]
	W		WEN_CH[110]		WEN_CH[108]		WEN_CH[106]		WEN_CH[104]			WEN_CH[101]	WEN_CH[100]	WEN_CH[99]	WEN_CH[98]	WEN_CH[97]	WEN_CH[96]
SARADCB_TCAWORR	R	AWOR_CH[127]	AWOR_CH[126]	AWOR_CH[125]	AWOR_CH[124]	0	AWOR_CH[122]	AWOR_CH[121]	AWOR_CH[120]	AWOR_CH[119]	AWOR_CH[118]	AWOR_CH[117]	AWOR_CH[116]	AWOR_CH[115]	AWOR_CH[114]	AWOR_CH[113]	AWOR_CH[112]
	W	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	0	AWOR_CH[110]	0	AWOR_CH[108]	0	AWOR_CH[106]	0	AWOR_CH[104]	0	0	AWOR_CH[101]	AWOR_CH[100]	AWOR_CH[99]	AWOR_CH[98]	AWOR_CH[97]	AWOR_CH[96]
	W		w1c		w1c		w1c		w1c			w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
SARADCB_TCDR96-101 TCDR104, TCDR106, TCDR108, TCDR110, TCDR112-122, TCDR124-127	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

Table 511. SARADC_B external channel register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ECIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	DSD											
	W																
SARADCB_ECIPR0	R	EOC_CH[159]	EOC_CH[158]	EOC_CH[157]	EOC_CH[156]	EOC_CH[155]	EOC_CH[154]	EOC_CH[153]	EOC_CH[152]	EOC_CH[151]	EOC_CH[150]	EOC_CH[149]	EOC_CH[148]	EOC_CH[147]	EOC_CH[146]	EOC_CH[145]	EOC_CH[144]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOC_CH[143]	EOC_CH[142]	EOC_CH[141]	EOC_CH[140]	EOC_CH[139]	EOC_CH[138]	EOC_CH[137]	EOC_CH[136]	EOC_CH[135]	EOC_CH[134]	EOC_CH[133]	EOC_CH[132]	EOC_CH[131]	EOC_CH[130]	EOC_CH[129]	EOC_CH[128]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECIPR1	R	EOC_CH[191]	EOC_CH[190]	EOC_CH[189]	EOC_CH[188]	EOC_CH[187]	EOC_CH[186]	EOC_CH[185]	EOC_CH[184]	EOC_CH[183]	EOC_CH[182]	EOC_CH[181]	EOC_CH[180]	EOC_CH[179]	EOC_CH[178]	EOC_CH[177]	EOC_CH[176]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOC_CH[175]	EOC_CH[174]	EOC_CH[173]	EOC_CH[172]	EOC_CH[171]	EOC_CH[170]	EOC_CH[169]	EOC_CH[168]	EOC_CH[167]	EOC_CH[166]	EOC_CH[165]	EOC_CH[164]	EOC_CH[163]	EOC_CH[162]	EOC_CH[161]	EOC_CH[160]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECIMR0	R	IM_CH[159]	IM_CH[158]	IM_CH[157]	IM_CH[156]	IM_CH[155]	IM_CH[154]	IM_CH[153]	IM_CH[152]	IM_CH[151]	IM_CH[150]	IM_CH[149]	IM_CH[148]	IM_CH[147]	IM_CH[146]	IM_CH[145]	IM_CH[144]
	W																
	R	IM_CH[143]	IM_CH[142]	IM_CH[141]	IM_CH[140]	IM_CH[139]	IM_CH[138]	IM_CH[137]	IM_CH[136]	IM_CH[135]	IM_CH[134]	IM_CH[133]	IM_CH[132]	IM_CH[131]	IM_CH[130]	IM_CH[129]	IM_CH[128]
	W																
SARADCB_ECIMR1	R	IM_CH[191]	IM_CH[190]	IM_CH[189]	IM_CH[188]	IM_CH[187]	IM_CH[186]	IM_CH[185]	IM_CH[184]	IM_CH[183]	IM_CH[182]	IM_CH[181]	IM_CH[180]	IM_CH[179]	IM_CH[178]	IM_CH[177]	IM_CH[176]
	W																
	R	IM_CH[175]	IM_CH[174]	IM_CH[173]	IM_CH[172]	IM_CH[171]	IM_CH[170]	IM_CH[169]	IM_CH[168]	IM_CH[167]	IM_CH[166]	IM_CH[165]	IM_CH[164]	IM_CH[163]	IM_CH[162]	IM_CH[161]	IM_CH[160]
	W																

[illegible]

Table 511. SARADC_B external channel register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ECJCMR1	R	JCE_CH[191]	JCE_CH[190]	JCE_CH[189]	JCE_CH[188]	JCE_CH[187]	JCE_CH[186]	JCE_CH[185]	JCE_CH[184]	JCE_CH[183]	JCE_CH[182]	JCE_CH[181]	JCE_CH[180]	JCE_CH[179]	JCE_CH[178]	JCE_CH[177]	JCE_CH[176]
	W																
	R	JCE_CH[175]	JCE_CH[174]	JCE_CH[173]	JCE_CH[172]	JCE_CH[171]	JCE_CH[170]	JCE_CH[169]	JCE_CH[168]	JCE_CH[167]	JCE_CH[166]	JCE_CH[165]	JCE_CH[164]	JCE_CH[163]	JCE_CH[162]	JCE_CH[161]	JCE_CH[160]
	W																
SARADCB_ECWSELR0	R	0	WSEL_CH135			0	WSEL_CH134			0	WSEL_CH133			0	WSEL_CH132		
	W																
	R	0	WSEL_CH131			0	WSEL_CH130			0	WSEL_CH129			0	WSEL_CH128		
	W																
SARADCB_ECWSELR1	R	0	WSEL_CH143			0	WSEL_CH142			0	WSEL_CH141			0	WSEL_CH140		
	W																
	R	0	WSEL_CH139			0	WSEL_CH138			0	WSEL_CH137			0	WSEL_CH136		
	W																
SARADCB_ECWSELR2	R	0	WSEL_CH151			0	WSEL_CH150			0	WSEL_CH149			0	WSEL_CH148		
	W																
	R	0	WSEL_CH147			0	WSEL_CH146			0	WSEL_CH145			0	WSEL_CH144		
	W																
SARADCB_ECWSELR3	R	0	WSEL_CH159			0	WSEL_CH158			0	WSEL_CH157			0	WSEL_CH156		
	W																
	R	0	WSEL_CH155			0	WSEL_CH154			0	WSEL_CH153			0	WSEL_CH152		
	W																
SARADCB_ECWSELR4	R	0	WSEL_CH167			0	WSEL_CH166			0	WSEL_CH165			0	WSEL_CH164		
	W																
	R	0	WSEL_CH163			0	WSEL_CH162			0	WSEL_CH161			0	WSEL_CH160		
	W																
SARADCB_ECWSELR5	R	0	WSEL_CH175			0	WSEL_CH174			0	WSEL_CH173			0	WSEL_CH172		
	W																
	R	0	WSEL_CH171			0	WSEL_CH170			0	WSEL_CH169			0	WSEL_CH168		
	W																

[illegible]

Table 511. SARADC_B external channel register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ECAWORR1	R	AWOR_CH[191]	AWOR_CH[190]	AWOR_CH[189]	AWOR_CH[188]	AWOR_CH[187]	AWOR_CH[186]	AWOR_CH[185]	AWOR_CH[184]	AWOR_CH[183]	AWOR_CH[182]	AWOR_CH[181]	AWOR_CH[180]	AWOR_CH[179]	AWOR_CH[178]	AWOR_CH[177]	AWOR_CH[176]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	AWOR_CH[175]	AWOR_CH[174]	AWOR_CH[173]	AWOR_CH[172]	AWOR_CH[171]	AWOR_CH[170]	AWOR_CH[169]	AWOR_CH[168]	AWOR_CH[167]	AWOR_CH[166]	AWOR_CH[165]	AWOR_CH[164]	AWOR_CH[163]	AWOR_CH[162]	AWOR_CH[161]	AWOR_CH[160]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECMICR0	R	0	ICSEL_ECH152_159							0	ICSEL_ECH144_151						
	W																
	R	0	ICSEL_ECH136_143							0	ICSEL_ECH128_135						
	W																
SARADCB_ECMICR1	R	0	ICSEL_ECH184_191							0	ICSEL_ECH176_183						
	W																
	R	0	ICSEL_ECH168_175							0	ICSEL_ECH160_167						
	W																
SARADCB_ECDR128-191	R	0	REFSEL	0	0	PCE	0	CTSEL	0	0	0	0	VALID	OVERW	RESULT		
	W																
	R	0	0	0	0	CDATA											
	W																

38.2.2.9.2 SARADC_0 register definitions

Table 512. SARADC_0 register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15			
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
SARADC0_MCR	R				0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0			
	W	OWREN	WLSIDE	MODE																
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	EDCSELF	PWDN			
	W																			

Table 512. SARADC_0 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC0_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC0_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC0_ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	EOC_CH[7]	EOC_CH[6]	EOC_CH[5]	EOC_CH[4]	0	0	0	0
	W									w1c	w1c	w1c	w1c				
SARADC0_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC0_ICIMR0	R																
	W																
	R	0	0	0	0	0	0	0	0	IM_CH[7]	IM_CH[6]	IM_CH[5]	IM_CH[4]	0	0	0	0
	W																
SARADC0_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 512. SARADC_0 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC0_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC0_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC0_ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DS_CH[7]	DS_CH[6]	DS_CH[5]	DS_CH[4]	0	0	0	0
	W																
SARADC0_WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC0_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC0_ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	NCE_CH[7]	NCE_CH[6]	NCE_CH[5]	NCE_CH[4]	0	0	0	0
	W																
SARADC0_ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	JCE_CH[7]	JCE_CH[6]	JCE_CH[5]	JCE_CH[4]	0	0	0	0
	W																

Table 512. SARADC_0 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC0_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC0_ICDR4-7	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC0_ICWSELR0	R	0	WSEL_CH7			0	WSEL_CH6			0	WSEL_CH5			0	WSEL_CH4		
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC0_ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WEN_CH[7]	WEN_CH[6]	WEN_CH[5]	WEN_CH[4]	0	0	0	0
	W																
SARADC0_ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	AWOR_CH[7]	AWOR_CH[6]	AWOR_CH[5]	AWOR_CH[4]	0	0	0	0
	W									w1c	w1c	w1c	w1c				

38.2.2.9.3 SARADC_1 register definitions

Table 513. SARADC_1 register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC1_MCR	R				0										0	0	0
	W	OWREN	WLSIDE	MODE		NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN		JEDGESEL	JTRGSEQ			
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC1_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC1_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC1_ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EOC_CH[15]	EOC_CH[14]	EOC_CH[13]	EOC_CH[12]	EOC_CH[11]	EOC_CH[10]	EOC_CH[9]	EOC_CH[8]	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c								
SARADC1_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC1_ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	IM_CH[15]	IM_CH[14]	IM_CH[13]	IM_CH[12]	IM_CH[11]	IM_CH[10]	IM_CH[9]	IM_CH[8]	0	0	0	0	0	0	0	0
	W																

Table 513. SARADC_1 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC1_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC1_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC1_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC1_ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DS_CH[15]	DS_CH[14]	DS_CH[13]	DS_CH[12]	DS_CH[11]	DS_CH[10]	DS_CH[9]	DS_CH[8]	0	0	0	0	0	0	0	0
	W																
SARADC1_WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC1_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC1_ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	NCE_CH[15]	NCE_CH[14]	NCE_CH[13]	NCE_CH[12]	NCE_CH[11]	NCE_CH[10]	NCE_CH[9]	NCE_CH[8]	0	0	0	0	0	0	0	0
	W																

Table 513. SARADC_1 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
SARADC1_ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	JCE_CH[15]	JCE_CH[14]	JCE_CH[13]	JCE_CH[12]	JCE_CH[11]	JCE_CH[10]	JCE_CH[9]	JCE_CH[8]	0	0	0	0	0	0	0	0	
	W																	
SARADC1_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	PDED								
	W																	
SARADC1_ICDR8-15	R	0	REFSEL	0	0	PCE	0	CTSEL	0	0	0	0	VALID	OVERW	RESULT			
	W																	
	R	0	0	0	0	CDATA												
	W																	
SARADC1_ICWSELR1	R	0	WSEL_CH15			0	WSEL_CH14			0	WSEL_CH13			0	WSEL_CH12			
	W																	
	R	0	WSEL_CH11			0	WSEL_CH10			0	WSEL_CH9			0	WSEL_CH8			
	W																	
SARADC1_ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	WEN_CH[15]	WEN_CH[14]	WEN_CH[13]	WEN_CH[12]	WEN_CH[11]	WEN_CH[10]	WEN_CH[9]	WEN_CH[8]	0	0	0	0	0	0	0	0	
	W																	
SARADC1_ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	AWOR_CH[15]	AWOR_CH[14]	AWOR_CH[13]	AWOR_CH[12]	AWOR_CH[11]	AWOR_CH[10]	AWOR_CH[9]	AWOR_CH[8]	0	0	0	0	0	0	0	0	
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c									

38.2.2.9.4 SARADC_2 register definitions

Table 514. SARADC_2 register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
SARADC2_MCR	R				0										0	0	0	
	W	OWREN	WLSIDE	MODE		NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN		JEDGESEL	JTRGSEQ				
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	EDCSELF	PWDN	
	W																	
SARADC2_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0	
	W																	
	R	CHADDR									0	0	0	0	0	ADCSTATUS		
	W																	
SARADC2_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH	
	W													w1c	w1c	w1c	w1c	
SARADC2_ICIPR0	R	0	0	0	0	0	0	0	0	EOC_CH[23]	EOC_CH[22]	0	0	EOC_CH[19]	EOC_CH[18]	EOC_CH[17]	EOC_CH[16]	
	W									w1c	w1c			w1c	w1c	w1c	w1c	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
SARADC2_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH	
	W																	
SARADC2_ICIMR0	R	0	0	0	0	0	0	0	0	IM_CH[23]	IM_CH[22]	0	0	IM_CH[19]	IM_CH[18]	IM_CH[17]	IM_CH[16]	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	

Table 514. SARADC_2 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC2_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC2_WTMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC2_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC2_ICDSR0	R	0	0	0	0	0	0	0	0	DS_CH[23]	DS_CH[22]	0	0	DS_CH[19]	DS_CH[18]	DS_CH[17]	DS_CH[16]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC2_WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC2_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC2_ICNCMR0	R	0	0	0	0	0	0	0	0	NCE_CH[23]	NCE_CH[22]	0	0	NCE_CH[19]	NCE_CH[18]	NCE_CH[17]	NCE_CH[16]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 514. SARADC_2 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
SARADC2_ICJCMR0	R	0	0	0	0	0	0	0	0	JCE_CH[23]	JCE_CH[22]	0	0	JCE_CH[19]	JCE_CH[18]	JCE_CH[17]	JCE_CH[16]	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
SARADC2_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	PDED								
	W																	
SARADC2_ICDR16-19, ICDR22-23	R	0	REFSEL	0	0	PCE	0	CTSEL	0	0	0	0	VALID	OVERW	RESULT			
	W																	
	R	0	0	0	0	CDATA												
	W																	
SARADC2_ICWSELR2	R	0	WSEL_CH23			0	WSEL_CH22			0	0	0	0	0	0	0	0	
	W																	
	R	0	WSEL_CH19			0	WSEL_CH18			0	WSEL_CH17			0	WSEL_CH16			
	W																	
SARADC2_ICWENR0	R	0	0	0	0	0	0	0	0	WEN_CH[23]	WEN_CH[22]	0	0	WEN_CH[19]	WEN_CH[18]	WEN_CH[17]	WEN_CH[16]	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
SARADC2_ICAWORR0	R	0	0	0	0	0	0	0	0	AWOR_CH[23]	AWOR_CH[22]	0	0	AWOR_CH[19]	AWOR_CH[18]	AWOR_CH[17]	AWOR_CH[16]	
	W									w1c	w1c			w1c	w1c	w1c	w1c	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	

38.2.2.9.5 SARADC_3 register definitions

Table 515. SARADC_3 register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
SARADC3_MCR	R				0										0	0	0	
	W	OWREN	WLSIDE	MODE		NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN		JEDGESEL	JTRGSEQ				
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	EDCSELF	PWDN	
	W																	
SARADC3_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0	
	W																	
	R	CHADDR									0	0	0	0	0	ADCSTATUS		
	W																	
SARADC3_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH	
	W													w1c	w1c	w1c	w1c	
SARADC3_ICIPR0	R	EOC_CH[31]	EOC_CH[30]	EOC_CH[29]	EOC_CH[28]	EOC_CH[27]	EOC_CH[26]	0	0	0	0	0	0	0	0	0	0	
	W	w1c	w1c	w1c	w1c	w1c	w1c											
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
SARADC3_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH	
	W																	
SARADC3_ICIMR0	R	IM_CH[31]	IM_CH[30]	IM_CH[29]	IM_CH[28]	IM_CH[27]	IM_CH[26]	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	

Table 515. SARADC_3 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC3_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC3_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC3_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC3_ICDSR0	R	DS_CH[31]	DS_CH[30]	DS_CH[29]	DS_CH[28]	DS_CH[27]	DS_CH[26]	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC3_WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC3_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC3_ICNCMR0	R	NCE_CH[31]	NCE_CH[30]	NCE_CH[29]	NCE_CH[28]	NCE_CH[27]	NCE_CH[26]	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 515. SARADC_3 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
SARADC3_ICJCMR0	R	JCE_CH[31]	JCE_CH[30]	JCE_CH[29]	JCE_CH[28]	JCE_CH[27]	JCE_CH[26]	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
SARADC3_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	PDED								
	W																	
SARADC3_ICDR26-31	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT		
	W																	
	R	0	0	0	0	CDATA												
	W																	
SARADC3_ICWSELR3	R	0	WSEL_CH31			0	WSEL_CH30			0	WSEL_CH29			0	WSEL_CH28			
	W																	
	R	0	WSEL_CH27			0	WSEL_CH26			0	0	0	0	0	0	0	0	
	W																	
SARADC3_ICWENR0	R	WEN_CH[31]	WEN_CH[30]	WEN_CH[29]	WEN_CH[28]	WEN_CH[27]	WEN_CH[26]	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
SARADC3_ICAWORR0	R	AWOR_CH[31]	AWOR_CH[30]	AWOR_CH[29]	AWOR_CH[28]	AWOR_CH[27]	AWOR_CH[26]	0	0	0	0	0	0	0	0	0	0	
	W	w1c	w1c	w1c	w1c	w1c	w1c											
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	

38.2.2.9.6 SARADC_4 register definitions

Table 516. SARADC_4 register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC4_MCR	R				0										0	0	0
	W	OWREN	WLSIDE	MODE		NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN		JEDGESEL	JTRGSEQ			
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC4_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC4_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC4_ICIPR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	EOC_CH[35]	EOC_CH[34]	EOC_CH[33]	EOC_CH[32]
	W													w1c	w1c	w1c	w1c
SARADC4_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC4_ICIMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	IM_CH[35]	IM_CH[34]	IM_CH[33]	IM_CH[32]
	W																

Table 516. SARADC_4 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
SARADC4_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L	
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
SARADC4_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L	
	W									MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L	
SARADC4_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN	
	W																	
SARADC4_ICDSR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	DS_CH[35]	DS_CH[34]	DS_CH[33]	DS_CH[32]	
	W													DS_CH[35]	DS_CH[34]	DS_CH[33]	DS_CH[32]	
SARADC4_WTHRHLR0–WTHRHLR3	R	0	0	0	0	THR _H												
	W																	
	R	0	0	0	0	THR _L												
	W																	
SARADC4_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	PRECHG					INPSAMP							
	W																	
SARADC4_ICNCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	NCE_CH[35]	NCE_CH[34]	NCE_CH[33]	NCE_CH[32]	
	W													NCE_CH[35]	NCE_CH[34]	NCE_CH[33]	NCE_CH[32]	

Table 516. SARADC_4 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
SARADC4_ICJCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	JCE_CH[35]	JCE_CH[34]	JCE_CH[33]	JCE_CH[32]	
	W																	
SARADC4_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	PDED								
	W																	
SARADC4_ICDR32-35	R	0	REFSEL	0	0	PCE	0	CTSEL	0	0	0	0	VALID	OVERW	RESULT			
	W																	
	R	0	0	0	0	CDATA												
	W																	
SARADC4_ICWSELR4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	WSEL_CH35			0	WSEL_CH34			0	WSEL_CH33			0	WSEL_CH32			
	W																	
SARADC4_ICWENR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	WEN_CH[35]	WEN_CH[34]	WEN_CH[33]	WEN_CH[32]	
	W																	
SARADC4_ICAWORR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	AWOR_CH[35]	AWOR_CH[34]	AWOR_CH[33]	AWOR_CH[32]	
	W																	
	W													w1c	w1c	w1c	w1c	

38.2.2.9.7 SARADC_6 register definitions

Table 517. SARADC_6 register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC6_MCR	R				0										0	0	0
	W	OWREN	WLSIDE	MODE		NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN		JEDGESEL	JTRGSEQ			
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC6_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC6_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC6_ICIPR1	R	0	0	0	0	0	0	0	0	0	0	0	0	EOC_CH[51]	EOC_CH[50]	EOC_CH[49]	EOC_CH[48]
	W													w1c	w1c	w1c	w1c
	R	0	0	0	0	EOC_CH[43]	EOC_CH[42]	EOC_CH[41]	EOC_CH[40]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								
SARADC6_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																

Table 517. SARADC_6 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15			
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
SARADC6_ICIMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	IM_CH[51]	IM_CH[50]	IM_CH[49]	IM_CH[48]			
	W																			
	R	0	0	0	0	IM_CH[43]	IM_CH[42]	IM_CH[41]	IM_CH[40]	0	0	0	0	0	0	0	0			
	W																			
SARADC6_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	W																			
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L			
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			
SARADC6_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	W																			
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L			
	W																			
SARADC6_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	W																			
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN			
	W																			
SARADC6_ICDSR1	R	0	0	0	0	0	0	0	0	0	0	0	0	DS_CH[51]	DS_CH[50]	DS_CH[49]	DS_CH[48]			
	W																			
	R	0	0	0	0	DS_CH[43]	DS_CH[42]	DS_CH[41]	DS_CH[40]	0	0	0	0	0	0	0	0			
	W																			
SARADC6_WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH														
	W																			
	R	0	0	0	0	THRL														
	W																			
SARADC6_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	W																			
	R	0	0	0	0	PRECHG				INPSAMP										
	W																			

Table 517. SARADC_6 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC6_ICNCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	NCE_CH[51]	NCE_CH[50]	NCE_CH[49]	NCE_CH[48]
	W																
	R	0	0	0	0	NCE_CH[43]	NCE_CH[42]	NCE_CH[41]	NCE_CH[40]	0	0	0	0	0	0	0	0
	W																
SARADC6_ICJCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	JCE_CH[51]	JCE_CH[50]	JCE_CH[49]	JCE_CH[48]
	W																
	R	0	0	0	0	JCE_CH[43]	JCE_CH[42]	JCE_CH[41]	JCE_CH[40]	0	0	0	0	0	0	0	0
	W																
SARADC6_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC6_ICDR40-43, ICDR48-51	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC6_ICWSELR5	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH43			0	WSEL_CH42			0	WSEL_CH41			0	WSEL_CH40		
	W																
SARADC6_ICWSELR6	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH51			0	WSEL_CH50			0	WSEL_CH49			0	WSEL_CH48		
	W																

Table 517. SARADC_6 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC6_ICWENR1	R	0	0	0	0	0	0	0	0	0	0	0	0	WEN_CH[51]	WEN_CH[50]	WEN_CH[49]	WEN_CH[48]
	W													WEN_CH[51]	WEN_CH[50]	WEN_CH[49]	WEN_CH[48]
	R	0	0	0	0	WEN_CH[43]	WEN_CH[42]	WEN_CH[41]	WEN_CH[40]	0	0	0	0	0	0	0	0
	W					WEN_CH[43]	WEN_CH[42]	WEN_CH[41]	WEN_CH[40]								
SARADC6_ICAWORR1	R	0	0	0	0	0	0	0	0	0	0	0	0	AWOR_CH[51]	AWOR_CH[50]	AWOR_CH[49]	AWOR_CH[48]
	W													w1c	w1c	w1c	w1c
	R	0	0	0	0	AWOR_CH[43]	AWOR_CH[42]	AWOR_CH[41]	AWOR_CH[40]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								

38.2.2.9.8 SARADC_7 register definitions

Table 518. SARADC_7 register definitions

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC7_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W	OWREN	WLSIDE	MODE		NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ			
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W					JTRGSEL				ABORT	ABORT		FRZ			EDCSELF	PWDN
SARADC7_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																

Table 518. SARADC_7 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC7_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC7_ICIPR1	R	0	0	0	0	EOC_CH[59]	EOC_CH[58]	EOC_CH[57]	EOC_CH[56]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC7_ICIMR1	R	0	0	0	0	IM_CH[59]	IM_CH[58]	IM_CH[57]	IM_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC7_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKW3H	MSKW3L	MSKW2H	MSKW2L	MSKW1H	MSKW1L	MSKW0H	MSKW0L
	W																

Table 518. SARADC_7 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC7_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC7_ICDSR1	R	0	0	0	0	DS_CH[59]	DS_CH[58]	DS_CH[57]	DS_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC7_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC7_ICNCMR1	R	0	0	0	0	NCE_CH[59]	NCE_CH[58]	NCE_CH[57]	NCE_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_ICJCMR1	R	0	0	0	0	JCE_CH[59]	JCE_CH[58]	JCE_CH[57]	JCE_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_PDED	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																

Table 518. SARADC_7 register definitions (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC7_ICDR56-59	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC7_ICWSELR7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH59			0	WSEL_CH58			0	WSEL_CH57			0	WSEL_CH56		
	W																
SARADC7_ICWENR1	R	0	0	0	0	WEN_CH[59]	WEN_CH[58]	WEN_CH[57]	WEN_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_ICAWORR1	R	0	0	0	0	AWOR_CH[59]	AWOR_CH[58]	AWOR_CH[57]	AWOR_CH[56]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

39 Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface

39.1 Introduction

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides accurate conversion data for a wide range of applications.

39.2 Overview

[Figure 453](#) depicts the block diagram of an SDADC and its digital control block. The SDADC digital interface block provides all control information for operating mode selection (single-ended or differential), analog input gain, decimation rate, high-pass filter enable, etc. It also generates the bias controls for common mode voltage selection, to bias the positive and negative terminals to different voltage levels (for example, half scale bias, reference ground, reference supply) for diagnostics. It generates the analog multiplexer control signals for the selected channels. It also provides external modulator support and keeps the internal modulator in low consumption mode. To assist synchronous operation across multiple ADC blocks, hardware trigger support is implemented to start data conversion.

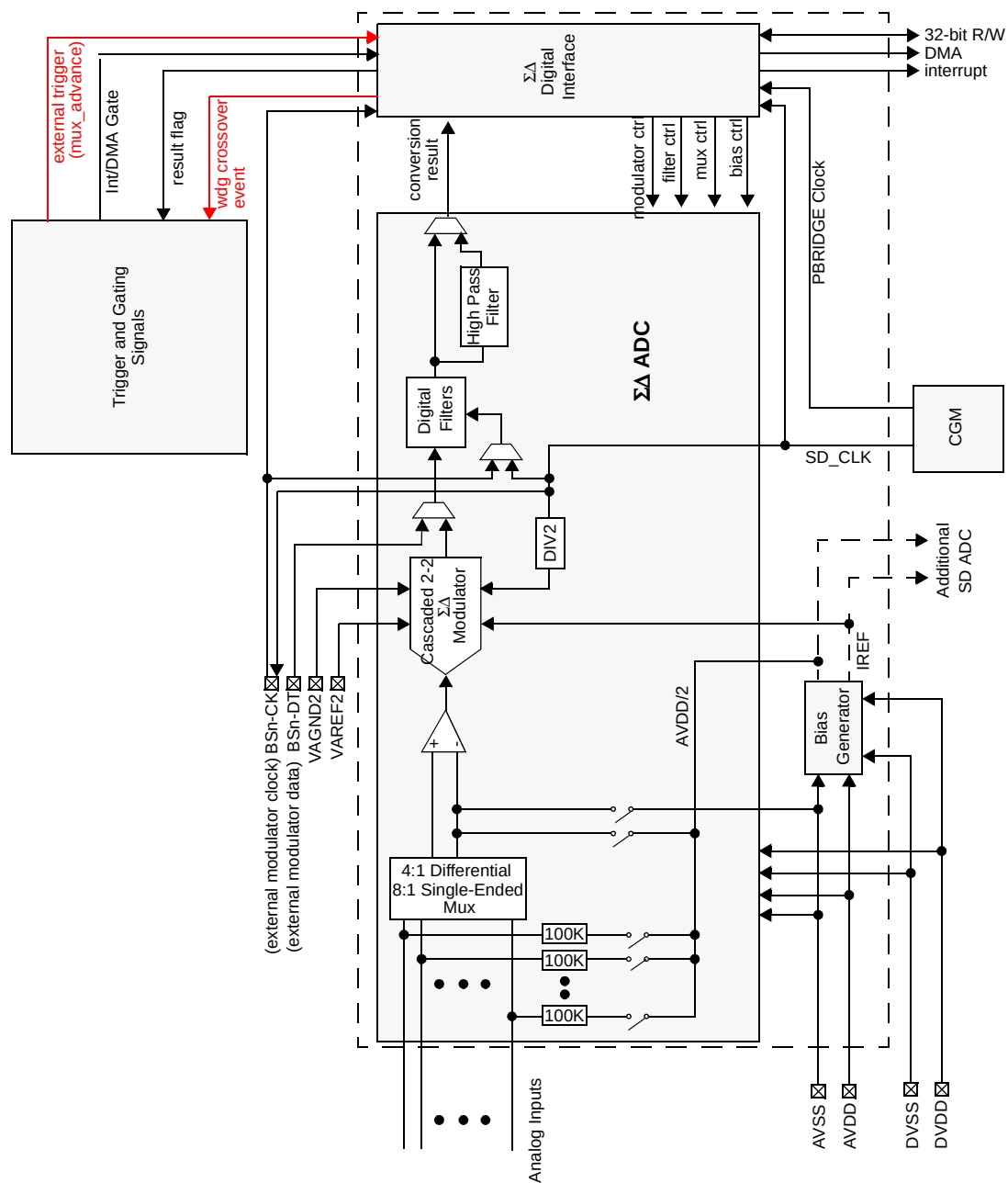


Figure 453. SDADC block diagram

39.3 Features

- 16-bit data resolution output
- Single-ended input or differential input mode of operation
- Programmable wraparound mechanism for both modes of operation
 - Configurable trigger sources: hardware or software
 - Configurable initial entry and wraparound values for the loop
- Configurable biasing for negative input terminal in single-ended mode
- Gain and offset calibration support using fixed bias for input terminals
- Programmable decimation rate
- Programmable gain for analog inputs
- Optional external modulator support
- Optional high-pass filter for pure AC application
- Hardware trigger support for synchronous operation of multiple SDADC blocks
- Trigger event output generation by software
- Programmable FIFO structure for storing 16-bit converted data
- Interrupt/DMA request generation based on various conditions:
 - Data buffer at or above threshold
 - Data buffer overrun
 - WDG crossover event
- Programmable threshold range WDG support
- Low consumption mode
- Optional conversion process halt mechanism on SoC debug request

39.4 Modes of operation

This section describes the different operation modes of the SDADC. When exiting reset, the mode of operation is determined by the Module Configuration Register (MCR).

39.4.1 Differential input mode

This mode is entered by negating the MCR[MODE] bit. In this mode, a pair of analog inputs are connected to positive and negative terminals of ADC modulator. In order to support gain and offset calibration for diagnostics, it is possible in this mode to select fixed bias voltages to both input terminals.

39.4.2 Single-ended input mode

This mode is entered by asserting the MCR[MODE] bit. In this mode, the negative input terminal is biased with a fixed voltage based on selection information in MCR, and the other analog input is connected to the selected external analog channel.

39.4.3 External modulator mode

This mode is entered by asserting the MCR[EMSEL] bit. In this mode, the data stream and clock outputs from the external modulator available on EMDATA/EMCLK input pins are routed directly to the SDADC block. The internal modulator is bypassed in this mode. Only

the digital filters of the SDADC will be used for result calculation. The source for the external modulator clock can be the internal on-chip clock source or the direct clock output coming from the external modulator. The external pin function control for these pins must be configured in the integration logic on a device, and is not included in the SDADC digital interface.

39.5 External signal description

[Table 519](#) shows the list of signals driven by external pins. At the chip integration level, some of the digital and analog signals might share pins or may not be available external to the chip.

Table 519. Signal properties

Name	Port	Function	I/O	Reset	Pull-up
AN[0:7]	SoC-specific	Single-ended analog inputs or Differential pairs of analog inputs	I	—	—
EMDATA	SoC-specific	External modulator data input	I	0	—
EMCLK	SoC-specific	External modulator clock input	I	0	—
VREFP	SoC-specific	Voltage reference for SDADC	I	—	—
VREFN	SoC-specific	Ground reference for SDADC	I	—	—
AVDD	SoC-specific	Analog voltage supply for SDADC	I	—	—
AVSS	SoC-specific	Analog ground supply for SDADC	I	—	—
DVDD	SoC-specific	Digital voltage supply for SDADC	I	—	—
DVSS	SoC-specific	Digital ground supply for SDADC	I	—	—

39.5.1 Detailed signal descriptions

[Table 520](#) provides a detailed description of the external signals.

Table 520. Detailed signal descriptions

Signal	I/O	Description
AN[0:7]	I	Each analog input can be used as a single-ended analog input or one of a differential pair of analog inputs which is connected to either positive or negative terminals of the SDADC, depending on the mode of operation and polarity selection.
		State meaning Analog input
		Timing Can be sampled at any time by the ADC.
EMDATA	I	Data stream input provided by external modulator.
		State meaning Asserted—Data input is 1 Negated—Data input is 0
		Timing Assertion—May occur at any time, but data stream is valid only when external modulator is selected by MCR[MODSEL] = 1. Negation—Ignored when external modulator is not selected.

Table 520. Detailed signal descriptions (continued)

Signal	I/O	Description
EMCLK	I	Clock input provided by external modulator.
		State meaning Asserted—clock high level Negated—clock low level
		Timing Assertion—May occur at any time, but clock is valid only when external modulator is selected by MCR[MODSEL] = 1. Negation—Ignored when external modulator is not selected.

39.6 Memory map and register descriptions

39.6.1 Memory map

This section provides the memory map for the SDADC block.

Table 521. SDADC Digital interface memory map

Offset or address	Register	Register classification ⁽¹⁾	Location
0x0000	SDADC_MCR – Module Configuration Register	Saf-Relv	on page 1252
0x0004	SDADC_CSR – Channel Selection Register	Saf-Relv	on page 1256
0x0008	SDADC_RKR – Reset Key Register	No-Saf	on page 1258
0x000C	SDADC_SFR – Status Flag Register	No-Saf	on page 1259
0x0010	SDADC_RSER – Request Select and Enable Register	Saf-Relv	on page 1260
0x0014	SDADC_OSDR – Output Settling Delay Register	Saf-Relv	on page 1262
0x0018	SDADC_FCR – FIFO Control Register	Saf-Relv	on page 1263
0x001C	SDADC_STKR – Software Trigger Key Register	No-Saf	on page 1264
0x0020	SDADC_CDR – Converted Data Register	No-Saf	on page 1265
0x0024	SDADC_WTHHLR – WDG Threshold Register	Saf-Relv	on page 1265

1. Saf-Relv = Safety Relevant, No-Saf = Not Safety Relevant.

39.6.2 Register descriptions

This section provides the block's register bit descriptions in address order.

39.6.2.1 Module Configuration Register (MCR)

The Module Configuration register (MCR) consists of different control bits to select the operating modes and various configuration settings which define the behavior of the SDADC block.

Figure 454. Module Configuration Register (MCR)

Offset 0x0000												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PDR					0	PGAN			0	0	EMSEL	HPFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDGEN	TRIGEDSEL	TRIGEN	TRIGSEL					FRZ	0	0	VCOMSEL	WRMODE	GECEN	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 522. MCR field descriptions

Field	Description
0–2	Reserved
3–7 PDR	Programmable Decimation Rate This field selects the over-sampling ratio to be applied to support different passbands with a fixed input sampling clock. The output data rate f_d is equal to $f_s / (2 \times \text{OSR})$, where f_s is the input sampling clock frequency. When external modulator is selected, the output data rate is independent of this field and is fixed to $f_s / 256$. 00000 OSR = 24 00001 OSR = 28 00010 OSR = 32 00011 OSR = 36 00100 OSR = 40 00101 OSR = 44 00110 OSR = 48 00111 OSR = 56 01000 OSR = 64 01001 OSR = 72 01010 OSR = 75 01011 OSR = 80 01100 OSR = 88 01101 OSR = 96 01110 OSR = 112 01111 OSR = 128 10000 OSR = 144 10001 OSR = 160 10010 OSR = 176 10011 OSR = 192 10100 OSR = 224 10101 OSR = 256 10110 Reserved 11111 Reserved
8	Reserved
9–11 PGAN	Programmable Gain This field selects the gain to be applied to the analog input stage of the SDADC. The effective analog input becomes the input voltage level multiplied by the gain factor. 000 Gain = 1 001 Gain = 2 010 Gain = 4 011 Gain = 8 100 Reserved 101 Reserved 110 Reserved 111 Gain = 16
12	Reserved
13	Reserved
14 EMSEL	External Modulator Selection 0 External modulator data and clock inputs are ignored 1 External modulator data stream and clock inputs are provided to SDADC

Table 522. MCR field descriptions (continued)

Field	Description
15 HPFEN	High Pass Filter Enable 0 High-pass (DC removal) filter is disabled 1 High-pass (DC removal) filter is enabled
16 WDGEN	Watchdog Enable This bit enables the WDG monitor. 0 WDG disabled 1 WDG is enabled
17–18 TRIGEDSEL	Trigger Edge Selection 00 Falling edge of trigger input is selected 01 Rising edge of trigger input is selected 1x Both edges of trigger input are selected
19 TRIGEN	Trigger Enable This field enables the hardware trigger input to initiate a fresh conversion. Upon receiving a trigger event, SDADC reset input is asserted and deasserted synchronously with respect to the peripheral clock. The reset pulse width is fixed to four peripheral clock cycles. The trigger event is ignored if the SDADC internal modulator is not enabled (MCR[EN] = 0) or the external modulator is selected (MCR[EMSEL] = 1). 0 Trigger input is disabled 1 Trigger input is enabled
20–23 TRIGSEL	Trigger Input Selection This field selects which input will be used for hardware-triggered conversions. 0000 Input trigger 0 is selected 0001 Input trigger 1 is selected ... 1111 Input trigger 15 is selected
24 FRZ	Freeze This field enables stopping the SDADC conversions at the end of the current channel conversion when the SoC enters debug mode. When the SoC enters debug mode, further conversions by the SDADC analog block will be stopped, and converted data output from the SDADC analog block are ignored. The ongoing channel conversion will be completed and the converted data is stored. When the SoC exits debug mode, SDADC resumes the job that was left before halting the conversions. 0 Conversions are not stopped 1 Conversions are stopped
25–26	Reserved
27 VCOMSEL	Common Voltage Bias Selection This field selects the common voltage bias for the negative input terminal of the SDADC during single-ended mode (MODE = 1). 0 Negative input terminal is biased with VREFN 1 Negative input terminal is biased with VREFP/2 (half-scale bias)
28 WRMODE	Wrap-Around Mode This bit selects the wraparound mechanism for conversion of programmed sequence of channels. 0 Wraparound mechanism disabled (in this case the default software control mechanism will be enabled) 1 Wraparound mechanism enabled

Table 522. MCR field descriptions (continued)

Field	Description
29 GECEN	Accurate Gain Error Mode Enable This bit selects the accurate gain error mode, and must be set during gain error calibration as well as during conversions requiring accurate gain error mode. It can be kept '0' if gain error calibration mode is not needed. 0 Gain error calibration mode disabled 1 Gain error calibration mode enabled
30 MODE	Mode selection 0 Differential input mode selected 1 Single-ended input mode selected
31 EN	Enable for SDADC block 0 SDADC internal modulator placed in low consumption mode 1 SDADC internal modulator enabled

39.6.2.2 Channel Selection Register (CSR)

The Channel Selection Register (CSR) selects analog inputs AN[0:7] to be connected to positive and negative terminals of the SDADC and controls the biasing of analog inputs to half-scale bias through 100 kΩ resistance.

Figure 455. Channel Selection Register (CSR)

Offset 0x0004												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	BIASEN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	ANCHSEL_WRAP			0	0	0	0	0	ANCHSEL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 523. CSR field descriptions

Field	Description
0–7	Reserved
8–15 BIASEN[7:0]	Bias enable If BIASEN[i] = 1, analog input AN[i] (could be single-ended input or differential analog input) is connected to half-scale bias through a 100 KΩ resistor.
16–20	Reserved

Table 523. CSR field descriptions (continued)

Field	Description
21–23 ANCHSEL_WRAP	Analog Channel Selection Wraparound Value (ANCHSEL_WRAP) When in wraparound mode, this field indicates the maximum value of the wraparound counter, after which it will wrap around back to initial value 000. This value should be programmed before enabling wraparound mechanism (MCR[WRMODE] = 1).
24–28	Reserved
29–31 ANCHSEL	Analog Channel Selection(ANCHSEL) Based on single-ended or differential mode of operation (MCR[MODE]), common mode voltage selection (MCR[VCOMSEL]), this field defines the connectivity of analog inputs to either positive or negative polarity terminals of the SDADC according to Table 524 . Wraparound Mode: – This field also indicates the initial entry value for the first loop of the wraparound sequence from where the wraparound counter will start incrementing when wraparound mode is entered (when wraparound mechanism enabled (MCR[WRMODE] = 1)). – If this field is written any time during wraparound mode (when MCR[WRMODE] = 1), the present value of the wraparound counter will be updated with the value of ANCHSEL — the next trigger will act on the updated channel, and the next increment will be on the updated channel. Here, the value of ANCHSEL chosen should be < ANCHSEL_WRAP; otherwise it will be ignored by hardware. – Wraparound will always be from initial counter value 000.

Table 524. Analog input AN[0:7] selection

MODE	VCOMSEL	ANCHSEL	INP (positive terminal)	INM (negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	

Table 524. Analog input AN[0:7] selection (continued)

MODE	VCOMSEL	ANCHSEL	INP (positive terminal)	INM (negative terminal)
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	AN[4]	AN[5]
		011	AN[6]	AN[7]
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

39.6.2.3 Reset Key Register (RKR)

The Reset Key Register (RKR) is the target for a predefined key write operation used to reset the SDADC to start a fresh conversion.

Figure 456. Reset Key Register (RKR)

Offset 0x0008												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESET_KEY															
W																
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Table 525. RKR field descriptions

Field	Description
0–15	Reserved
16–31 RESET_KEY[15:0]	Reset Key This field, when written with 0x5AF0, is used to generate the reset for the SDADC block and to reload the internal counter with the start value programmed in OSDR. Any write access to this register which is different from the predefined key is ignored. Read access will always return the inverted key.

39.6.2.4 Status Flag Register (SFR)

The Status Flag Register (SFR) contains status and flag bits. These bits are set by hardware and reflect the status of the SDADC and indicate the occurrence of events that may generate an interrupt or a DMA request. Software can clear some flags by writing '1' to those bits. Writing a '0' to a flag has no effect.

Figure 457. Status Flag Register (SFR)

Offset 0x000C												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	ANCHSEL_CNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	DFF	0	0	0	WTHH	WTHL	CVDF	DFORF	DFFF
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 526. SFR field descriptions

Field	Description
0–12	Reserved
13–15 ANCHSEL_CNT	Analog Channel Selection Counter(ANCHSEL_CNT) This field reflects the current value of the internal Analog Channel Selection Counter. It indicates the present number of the channel selected through the analog multiplexer.
16–22	Reserved
23 DFF	Data FIFO Empty Flag This field is asserted when there is no dataword in the FIFO. 0 Data FIFO is not empty 1 Data FIFO is empty
24–26	Reserved
27 WTHH	Watchdog Upper Threshold Cross Over Event This field is set when Converted Data crosses Upper Threshold of WDG. It can be cleared by software writing '1' to clear. 0 Watchdog Upper Threshold Cross Over Event did not occur 1 Watchdog Upper Threshold Cross Over Event occurred
28 WTHL	Watchdog Lower Threshold Cross Over Event This field is set when Converted Data crosses Lower Threshold of WDG. It can be cleared by software writing '1' to clear. 0 Watchdog Lower Threshold Cross Over Event did not occur 1 Watchdog Lower Threshold Cross Over Event occurred

Table 526. SFR field descriptions (continued)

Field	Description
29 CDVF	Converted Data Valid Flag This field gives the actual status of data coming from the SDADC block. The converted data is not valid or becomes corrupted whenever there is a change in configuration settings determined by MCR, channel selection programmed in CSR. Any write access to the registers MCR and CSR will reset this flag. This flag is set automatically after generating SDADC reset and internal timer counts OSD cycles of output clock f_d. 0 Data output from SDADC is not valid 1 Data output from SDADC is valid
30 DFORF	Data FIFO Overrun Flag This field indicates that software or DMA failed to prevent FIFO from overflowing with converted datawords. Software can clear this bit by writing '1' to it after reading the FIFO contents through CDR. This flag remains set if software or DMA failed to clear the condition that caused this flag setting. If DFORF is set, further datawords will not be received into FIFO even if sufficient room exists. 0 No overrun has occurred since the last time the flag was cleared 1 Overrun has occurred or the DFORF has not been cleared since the last overrun occurred
31 DFFF	Data FIFO Full Flag This bit is set when the number of converted datawords in the FIFO is equal to or more than the number indicated by FCR[FTHLD]. Software can clear this bit by writing '1' to it. This bit is also cleared by an acknowledgement from the DMA controller when DMA request generation is selected. This bit remains set if software or DMA failed to clear the condition that caused this flag setting. Even if DFFF is set, datawords will continue to be received until an overrun condition occurs. Datawords will not be received if the converted data is not valid (CDVF = 0). 0 The number of datawords in FIFO is less than the number indicated by FCR[FTHLD] 1 The number of datawords in FIFO is equal to or greater than the number indicated by FCR[FTHLD] at some point in time since this flag was last cleared

39.6.2.5 Request Select and Enable Register (RSER)

The Request Select and Enable Register (RSER) serves two purposes:

- Enables the flag bits in SFR to generate DMA requests or interrupt requests.
- Selects the type of request to generate.

Refer to the field descriptions for the type of requests that are supported.

Figure 458. Request Select and Enable Register (RSER)

Offset 0x0010

Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WTHDIRS	DFFDIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GDIGE	0	0	0	0	0	0	0	0	0	0	0	WTHDIRE	CDVEE	DFOIRE	DFFDIRE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 527. RSER field descriptions

Field	Description
0–13	Reserved
14 WTHDIRS	WDG Threshold Cross Over Event DMA/Interrupt Request Select When SFR[WTHH/WTHL] and RSER[WTHDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
15 DFFDIRS	Data FIFO Full DMA/Interrupt Request Select When the SFR[DFFF] and SFR[DFFDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
16 GDIGE	Global DMA/Interrupt Gating Enable This field determines whether all module interrupts and DMA requests need to be gated by an external gating signal or not. 0 No impact of external gating signal on module DMA/interrupt requests 1 All module DMA/interrupt requests are qualified only when external gating signal is asserted
17–27	Reserved
28 WTHDIRE	WDG Threshold Cross Over Event DMA/Interrupt Request Enable This field enables the SFR[WTHH/WTHL] field to generate a request. The WTHDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt/DMA request is disabled on WDG Threshold Cross Over Event 1 Interrupt/DMA request is enabled on WDG Threshold Cross Over Event

Table 527. RSER field descriptions (continued)

Field	Description
29 CDVEE	Conversion Data Valid Event Enable This field enables the SFR[CDVF] field to generate a hardware event to be used at SoC level. Once enabled, the event output will remain asserted as long as valid data conversions are ongoing. 0 Event output disabled 1 Event output assertion/deassertion based on the SFR[CDVF] field
30 DFORIE	Data FIFO Overrun Interrupt Enable This bit enables the SFR[DFORF] field to generate an interrupt request. The final interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt request is disabled when data FIFO overrun condition occurs 1 Interrupt request is enabled when data FIFO overrun condition occurs
31 DFFDIRE	Data FIFO Full DMA/Interrupt Request Enable This field enables the SFR[DFFF] field to generate a request. The DFFDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt/DMA request is disabled when data FIFO full condition occurs 1 Interrupt/DMA request is enabled when data FIFO full condition occurs

39.6.2.6 Output Settling Delay Register (OSDR)

The Output Settling Delay Register (OSDR) provides a delay value to qualify the converted output data coming from the SDADC.

Figure 459. Output Settling Delay Register (OSDR)

Offset 0x0014								Access: User Read/Write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	OSD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 528. OSDR field descriptions

Field	Description
0–23	Reserved
24–31 OSD[7:0]	Output Settling Delay This field defines the delay to qualify the conversion data stored in CDR. Whenever the SDADC block is reset in order to start the conversion from a fresh state, an internal timer is loaded with this start value. The counter counts down with output clock f_d until it reaches zero, and then it generates a flag which qualifies the converted data. The output clock f_d is equal to $f_s / (2 \times \text{OSR})$, where f_s is the input sampling clock frequency. When external modulator is selected, the output clock is fixed to $f_s / 256$.

39.6.2.7 FIFO Control Register (FCR)

The FIFO Control Register (FCR) provides the ability to turn on and off the FIFO functionality, and to select the size of FIFO depth. This register also provides the ability to set the programmable threshold for notification of dataword removal from the FIFO. This register may be read at any time, but should only be written when SDADC is disabled ($\text{MCR}[\text{EN}] = 0$).

Figure 460. FIFO Control Register (FCR)

Offset 0x0018												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	FTHLD				0	0	0	0	0	FSIZE		FE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 529. FCR field descriptions

Field	Description
0–19	Reserved
20–23 FTHLD	FIFO Threshold When the number of datawords in the data FIFO is equal to or greater than the value in FTHLD field, the FIFO full event is flagged. An interrupt or a DMA request will be generated as determined by DFFDIRE and DFFDIRS fields of RSER. For proper operation, the value in the FTHLD field must be set to be less than or equal to the FIFO size as indicated by FSIZE.
24–28	Reserved

Table 529. FCR field descriptions (continued)

Field	Description
29–30 FSIZE	FIFO Size The maximum number of converted datawords that can be stored in the data FIFO before an overrun occurs. This field is read-only. The reset value of this field depends on the implementation parameter to define the FIFO size. 00 FIFO depth = 1 dataword 01 FIFO depth = 4 datawords 10 FIFO depth = 8 datawords 11 FIFO depth = 16 datawords
31 FE	FIFO Enable When this field is set the built-in FIFO structure is enabled. The size of the FIFO structure is indicated by the FSIZE field. If this field is not set then the FIFO depth is set to one dataword regardless of the value in the FSIZE field. 0 Data FIFO is not enabled for multiple datawords; FIFO depth is one dataword 1 Data FIFO is enabled; FIFO depth is indicated by FSIZE

39.6.2.8 Software Trigger Key Register (STKR)

The Software Trigger Key Register (STKR) allows generating the trigger event output by a software write operation.

Figure 461. Software Trigger Key Register (STKR)

Offset 0x001C												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ST_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 530. STKR field descriptions

Field	Description
0–15	Reserved
16–31 ST_KEY[15:0]	Software Trigger Key This bitfield, when written with 0xFFFF, is used to generate a trigger event output which can be used to trigger conversions of multiple SDADC blocks synchronously depending on the SoC implementation. Any write access to this register which is different from the predefined key is ignored. Read access will always return 0x0000.

39.6.2.9 Converted Data Register (CDR)

The Converted Data Register (CDR) stores the converted data in the data FIFO.

Figure 462. Converted Data Register (CDR)

Offset 0x0020												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The conversion result in the CDATA word is a 16-bit, signed value.

Table 531. CDR field descriptions

Field	Description
0–15	Reserved
16–31 CDATA[15:0]	Converted Data Register The converted datawords can be read from FIFO by reading this register. The data width is 16 bits which has actual 16-bit data (bits 15 down to 0) coming from ADC.

39.6.2.10 WDG Threshold Register (WTHHLR)

This is the Converted Data Range WDG Threshold register.

Figure 463. WDG Threshold Register (WTHHLR)

Offset 0x0020												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	THRH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	THRL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 532. WTHHLR field descriptions

Field	Description
0–15 THRH	WDG Upper Threshold Value This indicates the upper threshold for the converted data on any channel, above which threshold a crossover event would be flagged through SFR[WTHH].
16–31 THRL	WDG Lower Threshold Value This indicates the lower threshold for the converted data on any channel, below which threshold cross over event would be flagged through SFR[WTHL].

Note: A single WDG monitor will check for lower and upper threshold crossover events.

39.7 Functional description

The SDADC has four distinct available modes determined by fields in MCR.

- Differential input mode
- Single-ended input mode
- External modulator mode
- Low consumption mode

39.7.1 Differential input mode

After system reset exit, this is the default mode of operation if SDADC is enabled by asserting MCR[EN]. This mode is entered by negating MCR[MODE]. In differential input mode, a pair of analog inputs among AN[0:7] is connected to the input terminals of the SDADC modulator block. The module supports four differential pairs: AN[0,1], AN[2,3], AN[4,5], and AN[6,7].

The modulator block has two input terminals, positive (INP) and negative (INM). The selection of input pair depends on the ANCHSEL field of CSR. The connectivity of each analog input in a given differential pair, whether it is connected to positive or negative

terminal, is determined according to [Table 524](#).

The differential input mode can also be used to perform gain and offset calibration, where the input terminals are biased with fixed voltages coming from the bias block.

39.7.2 Single-ended input mode

Single-ended input mode is entered by asserting MCR[MODE]. In this mode, one of the analog inputs among AN[0:7] is selected by the ANCHSEL field of CSR, and is connected to the positive terminal. The negative input terminal is biased with a fixed internal voltage. There are two internal bias voltages available: reference ground (VREFN) and half-scale bias (VREFP/2). The selection of this common mode voltage is controlled by MCR[VCOMSEL].

39.7.3 External modulator mode

This mode is selected by asserting the MCR[EMSEL] field. When external modulator is selected, the internal modulator can be put in low consumption mode by deasserting the MCR[EN] field. In this mode, the data stream and clock outputs from the external modulator available on EMDATA/EMCLK input pins are multiplexed with internal modulator data output and input sampling clock (f_s). External modulator data must be synchronized with respect to the falling edge of EMCLK. In this mode, hardware trigger input is ignored even if it is enabled by asserting the MCR[TRIGEN] field. All other configuration settings (for example gain, decimation rate, filter, etc.) programmed in MCR have no effect on the conversion.

39.7.4 Low consumption mode

This is the default mode after exiting from reset. This mode is exited by asserting the MCR[EN] field to enable the SDADC block. When data conversion is not required, the SDADC internal modulator can be put in low consumption mode to reduce power consumption.

39.7.5 Analog input multiplexing and bias support

The complete selection and connectivity of analog inputs AN[0:7] is based on the ANCHSEL field of CSR and the MODE and VCOMSEL bits of MCR, according to [Table 524](#). Each analog input can be biased to half scale ($AVDD/2$) via on-chip 100 k Ω resistance. This allows simple AC coupling to the external analog inputs through a capacitor. This biasing is applicable irrespective of whether analog channel is selected through multiplexer or not.

39.7.6 Programmable gain and decimation rate

All analog inputs can be configured to have a selectable input gain as defined in the PGAN field description in MCR. This means the input signal is sampled and the result is amplified by the factor determined by PGAN before providing the same to modulator.

The SDADC module is provided with a fixed input sampling clock f_s . The input signal is grossly oversampled by the modulator in order to reduce quantization noise. To support different passbands with a fixed sampling clock, a programmable decimation rate is implemented. The process of decimation, to eliminate redundant data at the output while retaining the necessary information, is controlled by the field PDR in MCR.

39.7.7 High-pass filter support

For pure AC applications, it is useful to remove any DC component in the input signal. An optional high-pass filter is implemented which can be enabled by asserting the MCR[HPFEN] field. The -3 dB frequency of the filter is fixed (not programmable) and is available in the device data sheet. This high-pass filter is implemented in the decimation filter logic of the SDADC and is applicable only to internal modulator mode.

39.7.8 Data conversion

Software Control Mechanism: The SDADC block will be in continuous conversion mode as soon as it is enabled. The conversion data is not accurate enough for some time, depending on the latency and output settling time specifications. This is true even when there is any change in the configuration settings provided by MCR, change in the analog channel selection due to mux switching time. To account for these delays, an internal timer is implemented which counts down from a start value determined by the OSD field of OSDR. This timer is reloaded whenever the RESET_KEY of RKR is written with 0x5AF0, which is required to start SDADC conversion from a fresh state with a changed configuration. The timer is stalled on reaching count '0' and asserts the SFR[CDVF] field. Once this flag is asserted, converted data is loaded into data FIFO on every rising edge of the output clock (f_d).

Any write access to the MCR and CSR registers will deassert the CDVF flag bit to indicate that data coming from SDADC is not valid or corrupted. If the CDVF flag is reset, the flags DFFF and DFORF are blocked from becoming asserted, i.e. transitioning from inactive state to active state. If DFFF bit is already set, this means the data FIFO is already full before CDVF deassertion.

The DFF flag bit (which not conditioned by the interrupt or DMA selection) is provided outside which can be routed to timer module at SoC level. **Wraparound Control Mechanism:** Here a wrap-around logic shall include all analog channels of the input multiplexer from Ain("000") to Ain(CSR[ANCHSEL_WRAP]) in both Single ended and Differential ended mode in wraparound sequence triggered by Harare or software source:

- MCR[WRMODE] bit would enable or disable the wraparound mechanism. Clearing the WRMODE bit automatically switches FSM back to Software control mode.
- Every valid trigger advances the wraparound counter to next channel to be converted.
- **Wraparound Counter:** An internal wraparound counter which points to analog channel to be converted takes the initial entry value for the first loop of execution from CSR[ANCHSEL] while entering wraparound mode. From next loop onwards the counter wraparounds to default initial value "000". Maximum value of the counter (wraparound value) should be programmed in CSR[ANCHSEL_WRAP] before entering wraparound mode, definition of ANCHSEL and ANCHSEL_WRAP is provided in [Section Table 524., Analog input AN\[0:7\] selection](#).
- Upon entering or during wraparound mode user should take care to program ANCHSEL value < ANCHSEL_WRAP value, otherwise hardware would reject the new ANCHSEL value.
- Possible trigger sources:
 - SOURCE1(Software Bit): writing to STKR.ST_KEY at any time during the wraparound mode (irrespective of MCR[TRIGEN or TRIGSEL] setting) generates a trigger pulse, each such trigger initiates a fresh conversion and advances the wraparound counter by one. Note: Care should be taken while programming the

- TRIGEN/TRIGSEL of each SDADC instance that no unintended simultaneous SW triggering of multiple instances happen.
- SOURCE2(Hardware Trigger): Here external trigger output (for e.g., mux_advance of GTM) enabled by MCR[TRIGEN or TRIGEDSEL or TRIGSEL] before entering wraparound mode for selective or all SDADC instances provides trigger, each such valid trigger initiates a fresh conversion and advances the wraparound counter by one. One of the hw_trig_in ports can be used to connect the external trigger output.
 - Next channel number to be converted can be updated by software even in the middle of wraparound sequence execution, this can be done by rewriting CSR[ANCHSEL]. After CSR[ANCHSEL] is rewritten, the next trigger takes updated value (wraparound counter would increment from the updated value).
 - Current wraparound counter value can be read from SFR[ANCHSEL_CNT] bitfield.

39.7.9 Hardware triggering

Software Control Mechanism: SDADC conversion can be triggered by various hardware events coming from either external pins or internal timers. The field TRIGSEL of MCR selects which hardware event is to be used for triggering conversions. This trigger event, which is synchronous to the peripheral clock, is enabled by asserting MCR[TRIGEN]. As soon as a trigger event is detected, reset input of SDADC block is asserted synchronously for four cycles. This will ensure that SDADC starts from a fresh state when reacting to the input trigger event. The SDADC internal modulator must be enabled before enabling the trigger event — otherwise the trigger event will be ignored. A hardware trigger event is also ignored when external modulator is selected (MCR[EMSEL] = 1).

If the same trigger event is provided to multiple SDADC digital interface modules and the respective enable controls (TRIGEN bit MCR) are asserted, synchronous operation of multiple SDADC blocks can be achieved with output results being updated at the same time. Once the reset input is deasserted, the first valid conversion output is stored in data FIFO after a configurable delay determined by OSDR[OSD].

It is also possible to generate a software trigger event output by a write access of STKR with a predefined key. This event output can be used as a trigger to achieve synchronous software start of multiple SDADC blocks.

In case of a hardware trigger, the skew/jitter will be less than 100 ns. It will be more than one ADC clock cycle due to synchronization. In case of a software trigger, all SDADCs start synchronous sampling.

Wraparound Control Mechanism: In wraparound control mode, hardware and software triggers are equally valid. The hardware trigger source should be selected before enabling the wraparound mode.

39.7.10 Interrupt/DMA request support

The SDADC has one condition that generates interrupts only and another that generates interrupts or DMA request.

The DFFF field is asserted when the data FIFO full condition is detected, indicating that the number of converted datawords stored in data FIFO is equal to or greater than the value indicated by threshold determined by FCR[FTHLD]. The interrupt or DMA request is generated only when RSER[DFFDIRE] is asserted. RSER[DFFDIRS] selects whether a DMA request or an interrupt request is generated.

The data FIFO overrun flag (DFORF) indicates the FIFO overflow condition. Further converted datawords cannot be received and data is lost. The interrupt request is generated only when RSER[DFORIE] is asserted.

Apart from above, a WDG threshold crossover event can also generate an interrupt or DMA request.

If RSER[GDIGE] is set, all module interrupt/DMA requests are qualified only when an external global gating signal is asserted.

39.7.11 Gain calibration support

To perform gain calibration, the following sequence needs to be applied.

1. Select differential mode of operation by writing MCR[MODE] to '0'.
2. Enable gain error calibration mode by writing MCR[GECEN] to '1'.
3. Configure the mux selection ANCHSEL field of CSR to '110'. This configuration will apply VREFP on positive terminal and VREFN on negative terminal.
4. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 0x00.
5. Disable the high-pass filter by deasserting MCR[HPFEN].
6. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
7. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 0b0111_1001_1001_1001, corresponding to full positive scale after attenuation inserted by the internal filter (1×0.95). The measurement should be repeated to reduce contribution of noise during calibration process. D_p is the average value of attenuated positive full scale given by $D_p = \text{AVERAGE}(\text{CDR}[\text{CDATA}])$.
8. Change the mux selection ANCHSEL field of CSR to '111'. This configuration will apply VREFN on positive terminal and VREFP on negative terminal.
9. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
10. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 0b1000_0110_0110_0110, corresponding to full negative scale after attenuation inserted by the internal filter (-1×0.95). The measurement should be repeated to reduce contribution of noise during calibration process. D_n is the average value of attenuated negative full scale given by $D_n = \text{AVERAGE}(\text{CDR}[\text{CDATA}])$.
11. The SDADC calibrated gain can be calculated as:

$$\text{Gain} = (D_p - D_n) / 2^{16}$$
12. The measured gain value can be used to nullify the gain errors in the digital output. For calibrated conversion, the data CDR[CDATA] provided by the SDADC should be normalized using the calculated gain: $\text{CDATA}_{\text{norm}}$ is given by $\text{CDATA}_{\text{norm}} = \text{CDR}[\text{CDATA}] / \text{Gain}$.

Note: The gain error calibration should only be done with gain equal to '1' (in other words, PGAN = 000). Nevertheless, the calculated gain can be applied to normalized data also for conversion with GAIN > 1.

Note: MCR[GECEN] = 1 ensures the accurate gain error mode is enabled. This bit should be set both during calibration process and after calibration, for application data conversion. Application data conversion must then be normalized using the measured gain.

Note: The higher the number of full-scale conversion (D_p , D_n) done, the higher the rejection of noise during the calibration. The number of conversion is dependent on application noise. It is recommended to run at least 16 conversions when calculating an average value.

39.7.12 Offset calibration support

To perform offset calibration, the following sequence must be applied.

1. Select differential mode of operation by writing MCR[MODE] to '0'.
2. Configure the mux selection ANCHSEL field of CSR to '100' or '101' as required.
 - a) CSR = '100' in case of data conversion after calibration in "single ended mode with negative input = $V_{SS_HV_ADR_D}$ "
 - b) CSR = '101' in case of data conversion after calibration in "differential mode" and "single ended mode with negative input = $(V_{DD_HV_ADR_D} - V_{SS_HV_ADR_D})/2$ "
3. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 0x00.
4. Disable the high pass filter by deasserting MCR[HPFEN].
5. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
6. Read the digital output stored in FIFO after the output settling time.
7. The measured offset can be used to nullify the offset error in the digital output. Expected output is 0b00_0000_0000_0000. The SDADC offset can be calculated as:
 Offset = Expected Output – Actual Output

Note: The offset must be calculated for each PGAN field setting since it is expected to vary with gain configuration of SDADC.

39.8 Initialization information

To initialize the SDADC registers for data conversion, the following sequence is required.

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter if required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC startup time or latency from reset exit.
6. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR. Otherwise, if data conversion need to be triggered by a hardware event, assert MCR[TRIGEN].
7. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

The following sequence needs to be followed when external modulator mode of SDADC is selected for data conversions.

1. Disable the SDADC internal modulator by deasserting MCR[EN].
2. Configure MCR to select the external modulator mode.
3. Select the required external pins EMDATA/EMCLK in the system integration logic (SIUL) of the device.
4. Configure OSD delay according to external modulator startup time or latency from reset exit.
5. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
6. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

40 Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface

40.1 Introduction

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for the application. It provides accurate and fast conversion data for a wide range of applications. Each SARADC analog block has its corresponding digital interface implemented at the SoC level.

40.2 Overview

The SARADC digital interface contains advanced features for normal or injected conversion modes of operation. The conversion can be triggered by software or hardware.

The digital interface can be configured to control up to 256 multiplexed analog input channels. There are three types of input channels:

- Internal channels, mapped to index ranging from 0 to 95
- Test channels, mapped to index ranging from 96 to 127
- External channels, mapped to index ranging from 128 to 255

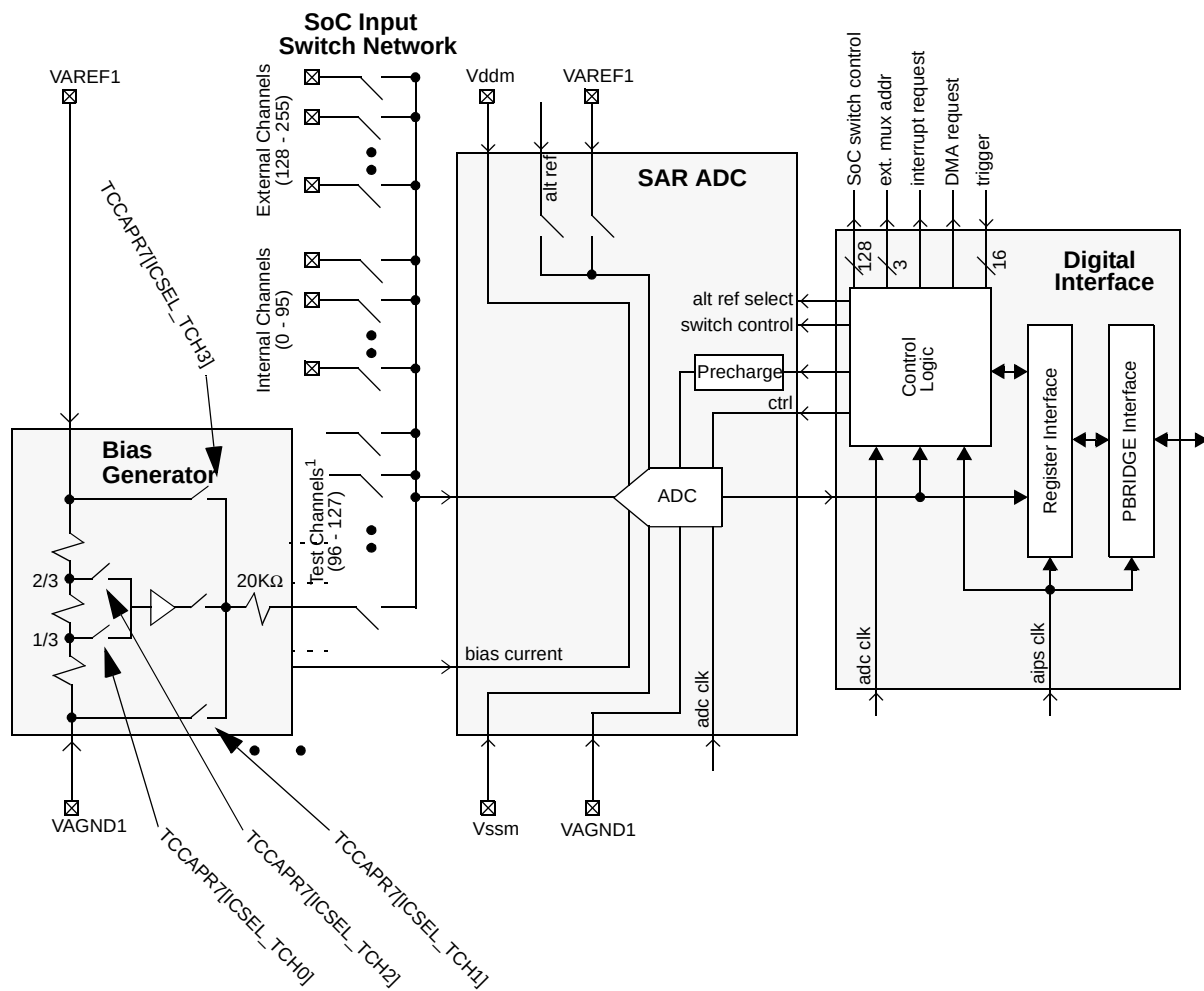
Note: The number of input channels and their mapping varies depending on your device configuration. See the device-specific Analog-to-Digital Converters (ADC) Configuration section (analog input pin multiplexing and SAR ADC channel assignment) for a channel mapping table.

The mask registers present within the digital interface can be properly programmed to configure which channel has to be converted. External channel selection is provided through external decode signals and are available as alternate functions on GPIO. Each set of eight external channels can be mapped to any internal analog channel by static programming by software in specific registers.

Four conversion timing registers exist that allow configuration of different precharging and sampling durations, and it is possible to select one of the conversion timing registers for each channel. For each channel, it is possible to choose the voltage reference through software. Analog Watchdogs allow continuous hardware monitoring of analog input channels. The digital interface also provides for interrupt/DMA support for various conditions related to end of channel conversions.

Figure 464 shows the SARADC block diagram.

Note: SARB implements a bias generator converting the V_{ddm} , $1/3 V_{ddm}$, $2/3 V_{ddm}$, and V_{ssm} reference points through the internal 20 k Ω source impedance.



1. Test channels are mapped to internal analog channels via the SARADCB_TCCAPRx registers

Figure 464. SARADC block diagram

40.3 Feature description

40.3.1 Main features

- Selectable 10-bit or 12-bit data resolution output
- Up to 96 internal channels, 32 test channels, 128 external channels supported; variable number of analog channels of each type controlled by parameters
- 4 different conversion timing registers selectable for any channel
- Mapping of external channel to any internal channel through static programming by software
- Shorting of test channel with internal channel through static programming by software
- External decode signals (3 signals) for selection of external analog mux inputs
- Normal conversion with One Shot/Scan modes
- Injected conversion with One Shot mode
- Normal conversion with dedicated trigger input
- Injected conversion with dedicated trigger input
- 2 different abort features that allow to abort either a single channel conversion or chain conversion
- Reference selection for each channel
- Power Down Mode
- Dedicated data register for each channel, containing the following information:
 - 10-bit or 12-bit conversion result in one half-word
 - Status byte which provides some conversion information such as mode of operation (Normal, Injected), data valid, data overwritten status
 - Control byte for reference selection, conversion timing parameter selection
- Configurable number of analog watchdogs
 - Trigger outputs on watchdog threshold crossover events
- Interrupt/DMA support for the following conditions
 - End of conversion of single channel for both normal, injected conversions
 - End of conversion chain for both normal, injected conversions
 - Watchdog thresholds crossover

40.4 Functional description

Two main conversion types are available within the SARADC digital interface:

- Normal conversion
- Injected conversion

40.4.1 Normal channel conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (ICNCMR0–2, TCNCMR, ECNCMR0–3). Each channel can be individually enabled by setting '1' in the corresponding field of these registers. Mask registers must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends (MSR[NSTART] bit is reset).

40.4.1.1 Start of normal conversion

The normal conversion can be started as follows:

- By software
 - The normal conversion chain starts when the MCR[NSTART] bit is set. It is recommended that the normal trigger enable MCR[NTRGEN] bit is reset during conversions started by software. Once normal conversion has started, any further write operation of MCR[NSTART] bit will have no effect during ongoing conversion, but the conversion will be enqueued after completion of current conversion in oneshot mode.
- By normal trigger
 - A normal trigger valid edge or level is detected to start the conversion. The normal trigger is enabled by setting the NTRGEN bit in MCR and two options are available. A programmed event (rising, falling or both edges depending on NEDGESEL bitfield of MCR) on the normal trigger input starts the normal conversion. Once normal conversion has started, any further valid trigger edge will be ignored during ongoing conversion. If the trigger mode is enabled (MCR[NTRGEN] bit is set), then software can only clear the MCR[NSTART] bit for terminating the conversion process. It is not possible to software to give any conversion request.

40.4.1.2 Normal conversion operating modes

Two operating modes are available for the normal conversion:

- One Shot Mode
- Scan Mode

To enter one of these modes, it is necessary to program the MCR[MODE] bit. The first phase of the conversion process involves sampling the analog channel and the next phase involves the conversion phase when the sampled analog value is converted to digital as shown in [Figure 465](#).

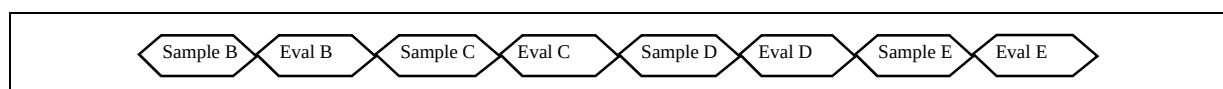


Figure 465. Normal conversion flow

In **One Shot Mode** (MODE = 0), a sequential conversion specified in the ICNCMR0–2, TCNCMR, ECNCMR0–3 mask registers is performed only once. At the end of each conversion, the digital result of the conversion is stored into the corresponding data register.

For Example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *One Shot Mode*. MODE = 0 is set for One Shot mode. Conversion starts from the channel B followed by conversion of channels D-E. At the end of conversion of channel E the scanning of channels stops.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time MCR[NSTART] bit is reset if the conversion is started by software.

Once the normal conversion operation starts in oneshot mode and another software conversion request is made writing the MCR[NSTART] bit and before MSR[NSTART] is cleared, one additional conversion is initiated with the same channel mask register settings after the current conversion is finished, this is called **enqueuing**. Enqueuing can help in

removing the delay between two consecutive oneshot chains and uncertainty caused by metastability between MCR[NSTART] bit written at system clock and generation of start pulse on ADC clock before every chain execution.

In **Scan Mode** (MODE = 1), a sequential conversion of N channels specified in the ICNCMR0–2, TCNCMR, ECNCMR0–3 registers is continuously performed. As in the previous case, at the end of each conversion the digital result of the conversion is stored into the corresponding data register.

In this mode, the software can start conversion by writing '1' to MCR[NSTART] bit or a valid normal trigger edge is detected which also sets MCR[NSTART] bit. The MCR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One Shot Mode, the MCR[NSTART] bit is not reset. It can be reset by software when the user needs to stop scan mode. In that case, the current scan conversion is completed and MCR[NSTART] bit also is reset after the last conversion of the chain.

For Example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *Scan Mode*. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B starts followed by conversion of the channels D-E. This sequence repeats itself till the MCR[NSTART] bit is reset by software.

In both the modes, at the end of each channel conversion an End Of Conversion interrupt is issued (if enabled by the corresponding mask bit) and at the end of the conversion sequence an End Of Chain interrupt is issued (if enabled by the corresponding mask bit). The End Of Chain interrupt is issued for each pass of scan mode.

40.4.2 Injected channel conversion

A conversion chain can be injected into the ongoing normal conversion configuring the injected mask registers (ICJCMR0–2, TCJCMR, ECJCMR0–3). As normal conversion, each internal, test or external channel can be individually selected. This injected conversion can only be in one-shot mode and interrupts the normal conversion. When an injected conversion is inserted, ongoing channel conversion is aborted and the injected channel request is processed. After the last channel in the injected chain is converted, normal conversion resumes from the channel at which the normal conversion was stopped as shown in the [Figure 466](#).

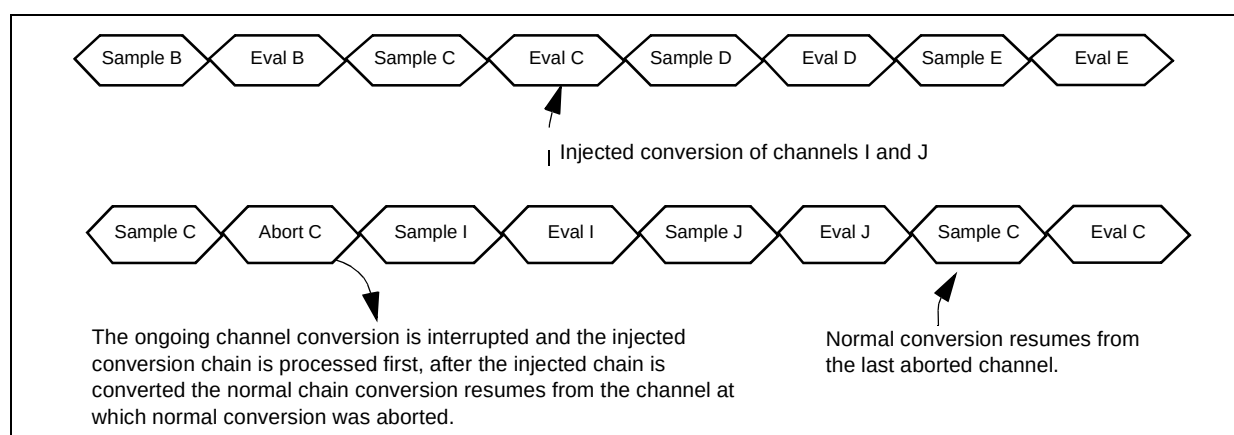


Figure 466. Injected sample/conversion sequence

The injected conversion can be started as follows:

- By software
 - The injected conversion chain starts when the MCR[JSTART] bit is set. The current conversion is suspended and the injected chain is converted. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed. It is recommended that the injection trigger enable MCR[JTRGEN] bit is reset during conversions started by software. Once injected conversion has started, any further write operation of MCR[JSTART] bit will not have any effect during ongoing conversion and the new request will get enqueued and executed after current injected chain finishes.
- By injection trigger
 - The injection trigger is enabled by setting the JTRGEN bit in MCR. A programmed event (rising, falling or both edges depending on JEDGESEL bitfield of MCR) on the injection external trigger starts the injected conversion and also sets the MSR[JSTART] bit. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed.
 - When JTRGSEQ bit is set, the injection trigger sequence feature is enabled. In this mode, each valid edge of injection trigger input will convert one analog channel. First valid edge will convert the first analog channel enabled through ICJCMR0–2, TCJCMR, ECJCMR0–3 registers. The next valid edge converts the next enabled analog input and so on. Once all the enabled channels are converted, the channel selection loops back to the first enabled channel for next trigger edge. If the JTRGSEQ is reset, the channel selection is reset again to first channel enabled through ICJCMR0–2, TCJCMR, ECJCMR0–3 registers. If JTRGSEQ is reset before completing the injected chain, the remaining channels are completed in one-shot after receiving next valid trigger edge.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time MSR[JSTART] is reset if the conversion is started by software.

At the end of each injected conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit) and at the end of the sequence an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit).

40.4.3 Abort conversion

Two different abort functions are provided:

- The user can abort the ongoing conversion by setting the ABORT bit in the MCR. The current conversion is aborted and the conversion of the next channel of the chain is immediately started (generating a new start pulse to the SARADC). In the case of an abort operation, the NSTART/JSTART bit remains set and the ABORT bit is reset after the conversion of the next channel starts. The EOC corresponding to the aborted channel is not generated. This behavior is true for normal or triggered/Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported generating an ECH interrupt.
- It is also possible to abort the current chain conversion by setting the ABORTCHAIN bit in the MCR. In that case the behavior of the ADC depends on the MODE bit. In fact, if scan mode is disabled, the NSTART bit is automatically reset together with the ABORTCHAIN bit. Otherwise, if the MODE bit is set to '1', a new chain conversion is

started. The EOC of the current aborted conversion is not generated but an ECH interrupt is generated to signal the end of the chain.

When a chain conversion abort is requested (ABORTCHAIN bit is set) while an injected conversion is running over a suspended Normal conversion, both injected chain and Normal conversion chain are aborted (both the NSTART and JSTART bits are also reset).

40.4.4 Analog conversion timings and reference selection

40.4.4.1 Conversion timings

In order to support different loadings and switching times (in particular for the external channel types) four different Conversion Timing registers are present (CTR0–3). Each conversion timing register contains PRECHG, INPSAMP bitfields to program the required duration for precharging and sampling phases. The selection of these registers for each channel is done by the SAMPSEL bitfield of corresponding channel data register.

Precharging phase allows to precharge or discharge the internal capacitor to a fixed internal voltage for duration determined by PRECHG bitfield prior to actual analog channel sampling start. This is useful for resetting information regarding the last converted data. Precharging feature can be enabled or disabled for each channel by programming the PCE bit of corresponding channel data register. If precharging is enabled for a given channel, the normal sequence of operation will be precharging, sampling followed by evaluation as shown in [Figure](#) . Please refer to device datasheet for the minimum precharge and sample durations required.

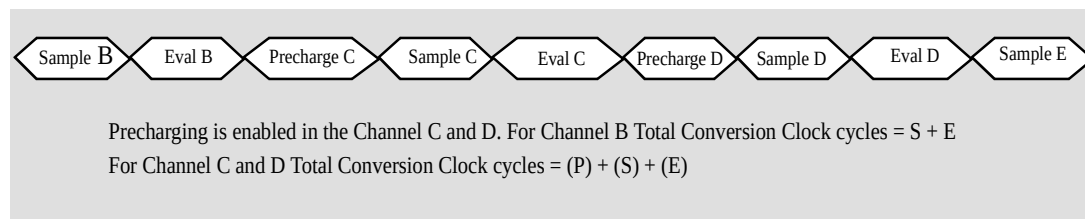


Figure 467. Analog conversion sequence

Bitfields PRECHG, INPSAMP are used to define the total conversion duration (t_{conv}) and in particular the partition among precharge duration (t_{prechg}), sampling phase duration (t_{sample}) and evaluation phase duration (t_{eval}).

The precharging phase duration is given by

$$t_{prechg} = \text{PRECHG} * t_{ck}$$

where PRECHG must be greater than or equal to 2 (hardware requirement) and PCE bit of channel data register is '1'. In case the value of PRECHG is found to be less than 2, it is automatically set to 2 inside SARADC. If the PCE bit is '0', the precharging phase is skipped and conversion starts with sampling phase directly.

The sampling phase duration is given by the following equation:

$$t_{sample} = \text{INPSAMP} * t_{ck}$$

where INPSAMP must be greater than or equal to 4 (hardware requirement). In case the value of INPSAMP is found to be less than 5, it is automatically set to 5 inside SARADC.

The total evaluation phase duration is given by the following equations:

$$t_{\text{eval}} = 25 * t_{\text{ck}}$$
$$t_{\text{eval}} = 21 * t_{\text{ck}} \text{ (for 10-bit conversion)}$$

In this phase, the internal sampling capacitor is disconnected from the analog channel and ADC estimates the digitized value of the sampled input using successive approximation algorithm. In the evaluation phase, all the bits are estimated sequentially to provide the conversion result.

The total conversion duration is (not including external multiplexing) is given by the following:

$$t_{\text{conv}} = t_{\text{prechg}} + t_{\text{sample}} + t_{\text{eval}}$$

The timings refer to the unit t_{ck} refers to reciprocal of f_{ck} , where f_{ck} = SARADC peripheral clock. Total conversion time is $< 5\mu\text{s}$.

Note: The current implementation of ADCDIG does not support hardware restriction over values of PRECHG and INPSAMP fields. As minimum values for these fields are product specific and need to be managed through software.

40.4.4.2 Alternate reference selection

In order to support different accuracy requirements, a voltage reference can be selected for each channel through REFSEL bit of the corresponding channel data register (for example, ICDRn). For some channels, this bit REFSEL is not software programmable and the selection configuration is fixed based on device-specific requirements.

40.4.5 Test channel connection with internal analog channel

The test channels are meant for monitoring various on-chip analog signals coming from temperature sensor, bandgap, voltage regulator and different voltage levels available on the chip. These analog signals can be interfaced to SARADC through test channels for testing purpose. Test channels are mapped from 96 to 127. Each test channel is controlled in the same way as normal internal channel.

SAR ADC Self Testing method: The test channel [127:124] is also mapped to various V_{ref} voltages (4 voltages) selection switches in the ADCBIAS block, the decoding of these channels to switch on and select a particular V_{ref} voltage is done in SoC level muxing logic. Refer ADC configuration for detailed mapping/control description of Self Test feature.

It is also possible to perform test channel conversions with and without shorting the normal analog channel pin with the internal test voltage. The test channel to internal channel mapping is enabled through top level synthesis parameter `testch_short_intch` vector, if shorting of a particular test channel has to be disabled then parameter should be set to '0' for corresponding test channel. This will result in removal of corresponding TCCAPR registers (see ADCDig IG for more details on `testch_short_intch`). For implemented TCCAPR registers to enable the short between test channel and any internal analog channel, `ESIC_TCHx` bit of TCCAPR0–7 registers should be set. The mapping of test channel to any internal analog channel is determined by the `ICSEL_TCHx` bitfield of TCCAPR0–7 registers.

For analog test channel assignment, refer to the device configuration or ADC configuration section.

After the conversion is performed on a test channel the converted data is stored at two CDR locations at the end of conversion, first location is the CDR belonging to the test channel on

which conversion is performed and second location is the CDR belonging to internal channel on which test channel is mapped.

40.4.6 External channel mapping to internal analog channel

External channels are supposed to be used in conjunction with a given internal analog channel along with an 8:1 external mux outside the device. Each input of the external mux can be mapped to a unique channel number, thus allowing it to perform conversions like any other channel of the ADC. External channels are mapped from 128 to 255.

External channels are implemented in the multiples of 8. Each slice of eight channels can be associated to any physical internal analog channel mapped from 0 to 95. This external-to-internal channel mapping is determined by the ICSEL_ECx_x bitfield of ECMICR0–3 registers. For example, when external multiplexer is planned to be used on internal channel 5, and the external inputs need to be mapped to channels 128 to 135, then this channel number 5 must be written to the ICSEL_ECH128_135 bitfield of the ECMICR0 register. After reset, all external channels are mapped to internal analog channel 0. If a conversion on a single external channel(out of group of 8) mapped to a particular internal channel has to be performed then only that particular channel should be enabled in the Conversion Mask Register (ECNMR or ECJCMR).

After the conversion is performed on a external channel the converted data is stored at two CDR locations at the end of conversion, first location is the CDR belonging to the external channel on which conversion is performed and second location is the CDR belonging to internal channel on which external channel is mapped.

40.4.7 Programmable analog watchdog

The analog watchdogs are used for determining whether the result of the conversion of a channel lies within a given guard area (as shown in the [Figure 468](#)) specified by an upper and a lower threshold value named THRH and THRL respectively.

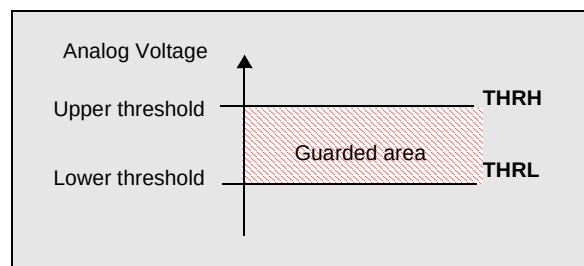


Figure 468. Analog watchdog guarded area

After the conversion of the selected channel a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guard area, then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the WTISR as explained in [Table 533](#). Depending on the MSKWDGxL and MSKWDGxH mask bits in the WTIMR, an interrupt is generated on threshold violation.

Table 533. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THR _H
0	1	converted data < THR _L
0	0	THR _H ≤ converted data ≤ THR _L

The lower and higher threshold values for the analog watchdog are programmed using THR_HL_R0–15 registers. Analog watchdog functionality for each channel can be enabled independently from ICWENR0–2, TCWENR, ECWEN0–3 registers and can select the watchdog threshold registers (THR_HL_R0–15) to be used by programming ICWSELR0–11, TCWSELR0–3, ECWSELR0–15 registers. The threshold registers selected by WSEL_CHx field of watchdog threshold selection registers will provide the threshold values.

For example, if channel 30 is to be monitored with the threshold values in register THR_HL_R12, then WSEL_CH30 is programmed to select THR_HL_R12 register to provide threshold values. The enabling is done by setting the bit corresponding to channel 30 in ICWENR0 register.

If the converted value for a particular channel lies outside the range specified by threshold values, then the corresponding bit is set in ICAWORR0–2, TCAWORR, ECAWORR0–3 registers. In this case, a set of threshold registers (THR_HL_R0–15) can be linked to several analog channels. The threshold values to be selected for a channel needs be programmed only once in ICWSELR, TCWSELR, ECWSELR registers.

There is an external trigger output port which provides a single clock cycle pulse to indicate WDG threshold crossover event on a particular WDG (up to 16 trigger outputs for up to 16 WDG instances).

Note: *.If the higher threshold for the analog watchdog is programmed lower than the lower threshold and the converted value is lesser than the lower threshold, then the WDGxL interrupt for the low threshold violation is set, else if the converted value is greater than the lower threshold (consequently also greater than the higher threshold) then the interrupt WDGxH for high threshold violation is set. Thus the user should take care of avoiding that situation as it could lead to misinterpretation of the watchdog interrupts.*

40.4.8 DMA functionality

A Direct Memory Access (DMA) request can be programmed after the conversion of every channel, by setting the respective masking bit in ICDSR0–2, TCDSR and ECDSR0–3 registers. The DMA masking registers must be programmed before starting any conversion. The DMA transfers can be enabled using the DMAE[DMAEN] bit. When the DMAE[DCLR] bit is set, the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

40.4.9 Interrupts

The SARADC digital interface generates the following maskable interrupt signals:

- End Of Conversion interrupts
 - NEOC (end of normal conversion of each channel)
 - NECH (end of normal chain)
 - JEOC (end of injected conversion of each channel)
 - JECH (end of injected chain)

The EOC_CH[x] bit of interrupt pending registers (ICIPR0–2, TCIPR, ECIPR0–3) is set when channel x completes the conversion (normal or injection). This pending status is qualified when the corresponding mask bit IM_CH[x] is also set in the interrupt mask registers (ICIMR0–2, TCIMR, ECIMR0–3).

The NEOC interrupt is generated only if the following conditions are met

- ISR[NEOC] bit is set after the normal conversion completion for any channel x.
- IMR[MSKNEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

The JEOC interrupt is generated only if the following conditions are met

- ISR[JEOC] bit is set after the injected conversion completion for any channel x.
- IMR[MSKJEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

It is recommended to clear the NEOC/JEOC/EOCTU bits of ISR and all the individual channel pending bits which are not masked while servicing the interrupt.

The ISR[NECH] bit is set when a normal channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

The ISR[JECH] bit is set when an injected channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

- WDGxL and WDGxH (Watchdog threshold) interrupt requests

The interrupts generated due to the analog watchdog are handled by registers WTISR and WTMR in order to check and enable the interrupt requests to external interrupt controller module. The watchdog interrupt source sets two pending bits WDGxH and WDGxL for each channel being monitored in the WTISR. The interrupt request is generated if at least one of the following conditions are met

- WTISR[WDGxL] bit is set and the corresponding mask bit MSKWDGxL is also set in WTMR for at least one channel.
- WTISR[WDGxH] bit is set and the corresponding mask bit MSKWDGxH is also set in WTMR for at least one channel.

The ISR and WTISR contain the interrupt pending request status flags. In case the user wants to clear a particular interrupt event status, writing a '1' to the corresponding status bit will clear the pending interrupt flag. (At this write operation, all other bits of the ISR must be maintained at '0'.)

The ADC module provides dedicated interrupt outputs for all interrupt sources. Also, all interrupt lines are OR-ed together to provide a single output to the external interrupt controller if a combined interrupt is required.

40.4.10 External decode signals selection and delay

The ADC provides three external decode signals used as select lines for 8:1 external mux. For each set of 8 external channels (for example [CH128:CH135], [CH136:CH143], and so on), the same 3-bit decode signals output will be used to select the inputs of external mux. These external mux select lines change in grey code sequence for each increment of channel selection. This ensures that only one line changes and avoids any current spikes or leakage if more than one line changes simultaneously.

In order to take into account the control switching time of the external analog mux, a Decode Signals Delay register is provided. Writing that register allows programming of the delay between the decoding signal selection and the actual start of conversion.

40.4.11 Power down mode

The SARADC analog block can be put in low current consumption mode by setting the PWDN bit in MCR. This is the default mode after exiting from reset. This state must be exited before starting any operation by resetting the PWDN bit in the MCR. When in powerdown mode, no conversion can be started. Bit ADCSTATUS[0] in the MSR is set only when ADC enters Powerdown mode.

The power-down mode can be requested at any time by setting the PWDN bit in the MCR. If a conversion is ongoing, the SARADC cannot immediately enter the power-down mode. In fact, the SARADC enters power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN bit.

After the power down phase is completed the process ongoing before the power down phase must be restarted manually (by setting the appropriate START bit). Resetting PWDN bit and setting NSTART or JSTART bit during the same cycle is forbidden.

40.5 Memory map and registers

This section provides memory map for the SARADC digital interface.

Table 534. SARADC digital interface register map

Offset	Register	Register classification ⁽¹⁾	Location
0x000	Main Configuration Register (MCR)	Saf-Relv	on page 1287
0x004	Main Status Register (MSR)	No-Saf	on page 1290
0x008–0x00C	Reserved		
0x010	Interrupt Status Register (ISR)	No-Saf	on page 1291
0x014–0x01C	Internal channel Interrupt Pending Registers 0–2 (ICIPR0–ICIPR2)	Saf-Relv	on page 1292
0x020	Interrupt Mask Register (IMR)	Saf-Relv	on page 1294
0x024–0x02C	Internal Channel Interrupt Mask Registers 0–2 (ICIMR0–ICIMR2)	Saf-Relv	on page 1295
0x030	Watchdog Threshold Interrupt Status Register (WTISR)	Saf-Relv	on page 1296
0x034	Watchdog Threshold Interrupt Mask Register (WTIMR)	Saf-Relv	on page 1297

Table 534. SARADC digital interface register map (continued)

Offset	Register	Register classification ⁽¹⁾	Location
0x038–0x03C	Reserved		
0x040	DMA Enable Register (DMAE)	Saf-Relv	on page 1298
0x044–0x04C	Internal Channel DMA Select Registers 0–2 (ICDSR0–ICDSR2)	Saf-Relv	on page 1299
0x050–0x05C	Reserved		
0x060–0x06C	Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)	Saf-Relv	on page 1300
0x070–0x090	Reserved		
0x094–0x0A0	Conversion Timing Registers 0–3 (CTR0–3)	Saf-Relv	on page 1301
0x0A4–0x0AC	Internal Channel Normal Conversion Mask Registers 0–2 (ICNCMR0–ICNCMR2)	Saf-Relv	on page 1302
0x0B0	Reserved		
0x0B4–0x0BC	Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–ICJCMR2)	Saf-Relv	on page 1303
0x0C0–0x0C4	Reserved		
0x0C8	Power Down Exit Delay Register (PDEDRE)	Saf-Relv	on page 1304
0x0CC–0x0FC	Reserved		
0x100–0x27C	Internal Channel Data Registers 0–95 (ICDR0–ICDR95)	Saf-Relv	on page 1304
0x280–0x2AC	Watchdog Threshold Registers 4-15 (WTHRHLR4–WTHRHLR15)	Saf-Relv	on page 1306
0x2B0–0x2DC	Internal Channel Watchdog Selection Registers 0–11 (ICWSELR0–ICWSELR11)	Saf-Relv	on page 1307
0x2E0–0x2E8	Internal Channel Watchdog Enable Registers 0–2 (ICWENR0–ICWENR2)	Saf-Relv	on page 1308
0x2EC	Reserved		
0x2F0–0x2F8	Internal Channel Analog Watchdog Out of Range registers 0–2 (ICAWORR0–ICAWORR2)	No-Saf	on page 1309
0x2FC–0x3FC	Reserved		
0x400	Test Channel Interrupt Pending Register (TCIPR)	Saf-Relv	on page 1311
0x404	Test Channel Interrupt Mask Register (TCIMR)	Saf-Lat	on page 1312
0x408	Test Channel DMA Select Register (TCDSR)	Saf-Lat	on page 1313
0x40C	Test Channel Normal Conversion Mask Register (TCNCMR)	Saf-Lat	on page 1314
0x410	Test Channel Injected Conversion Mask Register (TCJCMR)	Saf-Lat	on page 1315
0x414–0x420	Test Channel Watchdog Selection Registers 0–3 (TCWSELR0–TCWSELR3)	Saf-Lat	on page 1315
0x424	Test Channel Watchdog Enable Register (TCWENR)	Saf-Relv	on page 1317
0x428	Test Channel Analog Watchdog Out of Range Register (TCAWORR)	Saf-Relv	on page 1318
0x42C	Reserved		

Table 534. SARADC digital interface register map (continued)

Offset	Register	Register classification ⁽¹⁾	Location
0x430–0x44C	Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)	Saf-Relv	on page 1318
0x450–0x4CC	Test Channel Data Registers 96-127 (TCDR96–TCDR127)	Saf-Relv	on page 1320
0x4D0–0x4FC	Reserved		
0x500	External Channel Decode Signals Delay Register (ECDSDR)	Saf-Relv	on page 1322
0x504–0x50C	Reserved		
0x510–0x51C	External Channel Interrupt Pending Registers 0–3 (ECIPR0–ECIPR3)	Saf-Relv	on page 1323
0x520–0x52C	External Channel Interrupt Mask Registers 0–3 (ECIMR0–ECIMR3)	Saf-Relv	on page 1324
0x530–0x53C	External Channel DMA Select Registers 0–3 (ECDSR0–ECDSR3)	Saf-Relv	on page 1325
0x540–0x54C	External Channel Normal Conversion Mask Registers 0–3 (ECNCMR0–ECNCMR3)	Saf-Relv	on page 1326
0x550–0x55C	External Channel Injected Conversion Mask Registers 0–3 (ECJCMR0–ECJCMR3)	Saf-Relv	on page 1327
0x560–0x59C	External Channel Watchdog Selection Registers 0–15 (ECWSELR0–ECWSELR15)	Saf-Relv	on page 1328
0x5A0–0x5AC	External Channel Watchdog Enable Registers 0–3 (ECWENR0–ECWENR3)	Saf-Relv	on page 1329
0x5B0–0x5BC	External Channel Analog Watchdog Out of Range registers 0–3 (ECAWORR0–ECAWORR3)	Saf-Relv	on page 1330
0x5C0–0x5CC	External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–ECMICR3)	Saf-Relv	on page 1331
0x5D0–0x7CC	External Channel Data Registers 128-255 (ECDR128–ECDR255)	Saf-Relv	on page 1332

1. No-Saf = Not Safety Relevant, Saf-Relv = Safety Relevant, Saf-Lat = Safety Latent

40.5.1 Register descriptions

The following sections detail the individual registers in the SAR ADC digital interface.

40.5.1.1 Main Configuration Register (MCR)

Figure 469. Main Configuration Register (MCR)

Offset 0x000				Access: User Read/Write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	OWREN	WLSIDE	MODE	XSTRTEN ⁽¹⁾	NSTART	NTRGEN ⁽²⁾	NEDGESEL ⁽²⁾		JSTART	JTRGEN ⁽³⁾	JEDGESEL ⁽³⁾		JTRGSEQ ⁽³⁾	0	CTUEN ⁽⁴⁾	CTU_MODE ⁽⁴⁾			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	WTRIGOUT	0	0	0	JTRGSEL ⁽³⁾				ABORT CHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN			
W																			
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			

1. Only available with external start feature.
2. Only available with normal trigger feature.
3. Only available with injection trigger feature.
4. Only available with cross-triggering unit (CTU) feature.

Table 535. MCR field descriptions

Field	Description
0 OWREN	Overwrite enable This bit enables or disables the functionality to overwrite unread converted data in ICDR, TCDR, ECDR registers. 0 Prevents overwrite of unread converted data, new result is discarded 1 Enables converted data to be overwritten by a new conversion
1 WLSIDE	Write left/right-aligned 0 The conversion data in ICDR, TCDR, ECDR registers is written right-aligned. 1 Conversion data is left-aligned (from 15 to (15 – resolution + 1)).
2 MODE	One Shot/Scan 0 One Shot Mode—Configures the normal conversion of one chain. 1 Scan Mode—Configures continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.
3	Reserved
4 NSTART	Normal Start conversion Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation. 0 Causes the current chain conversion to finish and stops the operation 1 Starts the chain or scan conversion

Table 535. MCR field descriptions (continued)

Field	Description
5 NTRGEN	Normal trigger enable 0 Normal trigger disabled to start a normal conversion. 1 Normal trigger enabled to start a normal conversion.
6–7 NEDGESEL[1:0]	Normal trigger edge selection 00 Falling edge of normal trigger is selected. 01 Rising edge of normal trigger is selected. 1x Both rising and falling edges of normal trigger are selected.
8 JSTART	Injected Start conversion Setting this bit will start the configured injected analog channels to be converted by software. Resetting this bit has no effect, as the injected chain conversion cannot be interrupted. 0 Writing '0' has no effect. 1 Starts the injected chain conversion
9 JTRGEN	Injection trigger enable 0 Injection trigger disabled for channel injection (injected conversion cannot be started using an injection trigger) 1 Injection trigger enabled for channel injection
10–11 JEDGESEL[1:0]	Injection trigger edge selection 00 Falling edge of injection trigger is selected. 01 Rising edge of injection trigger is selected. 1x Both rising and falling edges of injection trigger are selected.
12 JTRGSEQ	Injection trigger sequence enable 0 Injection trigger sequence mode is disabled. 1 Injection trigger sequence mode is enabled.
13	Reserved Write of any value has no effect; read value is always 0.
14	Reserved
15	Reserved
16 WTRIGOUT	WDG Crossover External Trigger Output This control bit enables the Trigger on every threshold crossover event (crossing of Lower/Upper thresholds) of each WDG monitor to external trigger port (up to 16 ports for up to 16 WDGs). 0 Trigger Outputs disabled 1 Trigger Outputs enabled
17–19	Reserved Write of any value has no effect; read value is always 0.
20–23 JTRGSEL[1:0]	Injection Trigger Input Selection This bitfield selects which trigger input to be selected to start injected conversion. 0000 Trigger input 0 is selected 0001 Trigger input 1 is selected ... 1111 Trigger input 15 is selected.

Table 535. MCR field descriptions (continued)

Field	Description
24 ABORTCHAIN	Abort Chain When this bit is set, the ongoing Chain Conversion is aborted. This bit is reset by hardware as soon as a new conversion is requested. 0 Conversion is not affected 1 Aborts the ongoing chain conversion Note: Behaviour of this request during CTU mode is mentioned in CTU trigger/control mode section.
25 ABORT	Abort Conversion When this bit is set, the ongoing conversion is aborted and a new conversion is invoked. This bit is reset by hardware as soon as a new conversion is invoked. 0 Conversion is not affected 1 Aborts the ongoing conversion Note: Behaviour of this request during CTU mode is mentioned in CTU trigger/control mode section.
26	Reserved
27 FRZ	Freeze This bit enables to stop the SARADC conversions at the end of current channel conversion when SoC enters debug mode. 0 Conversions are not stopped. 1 When SoC enters debug mode, further conversions by SARADC analog block will be stopped. The ongoing channel conversion will be completed and the converted data is stored. When the SoC exits debug mode, SARADC resumes the job that was left before halting the conversions. Changing configuration registers during halted condition may cause a pending operation to fail when normal operation is resumed.
28–29	Reserved
30 EDCSELF	External Decode Channel Format Select (EDCSELF) This bit selects the format for 3-bit output port ipp_decode_extch used as select line for external decode channel 8:1 muxes. The last three bits of external channel number are passed as is (if value 0) or converted to gray code (if value 1) and passed to this port. 0 Binary format. 1 Gray Code format.
31 PWDN	Power-down enable When this bit is set, the analog module is requested to enter Power Down mode. When SARADC status is PWDN, resetting this bit starts SARADC transition to IDLE mode. 0 SARADC is in normal mode 1 SARADC has been requested to power down

40.5.1.2 Main Status Register (MSR)

Figure 470. Main Status Register (MSR)

Offset 0x004								Access: User Read Only								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTU START ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHADDR								0	0	0	0	0	ADCSTATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Only available with cross-triggering unit (CTU) feature.

Table 536. MSR field descriptions

Field	Description
0–3	Reserved Write of any value has no effect; read value is always 0.
4 NSTART	This status bit is used to signal that a normal conversion is ongoing. 0 Normal conversion is not taking place. 1 Normal conversion is ongoing or the normal conversion is pending due to injected conversion or CTU triggered injected conversion.
5–7	Reserved Write of any value has no effect; read value is always 0.
8 JSTART	This status bit is used to signal that an injected conversion is ongoing. 0 Injected conversion is not taking place. 1 Injected conversion is ongoing.
9–12	Reserved Write of any value has no effect; read value is always 0.
13 JABORTCHAIN	This status bit is used to signal that an Injected conversion chain has been aborted. This bit is reset when a new conversion starts. 0 Last injected conversion chain has not been aborted. 1 Last injected conversion chain has been aborted.
14	Reserved Write of any value has no effect; read value is always 0.
15	Reserved

Table 536. MSR field descriptions (continued)

Field	Description
16–23 CHADDR[7:0]	Channel under measure address This status bit is used to signal which channel is under measure.
24–28	Reserved
29–31 ADCSTATUS[2:0]	The value of this parameter depends on ADC status. While requesting an external channel conversion, SARADC digital interface needs to wait for a fixed time determined by DSD bitfield of DSDR before proceeding to actual sampling phase. This is known as wait state. 000 IDLE 001 Power-down 010 Wait state 011 — 100 Sample 101 — 110 Conversion 111 —

40.5.1.3 Interrupt Status Register (ISR)**Figure 471. Interrupt Status Register (ISR)**

Offset 0x010

Access: User Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR ⁽¹⁾	EOCTU ⁽²⁾	JEOC	JECH	NEOC	NECH
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only available with cross-triggering unit (CTU) feature.

2. Only available with cross-triggering unit (CTU) feature.

Table 537. ISR field descriptions

Field	Description
0–25	Reserved Write of any value has no effect; read value is always 0.
26	Reserved
27	Reserved
28 JEOC	End of injected channel conversion interrupt flag This bit indicates the end of conversion for an injected channel. 0 End of conversion of injected channel has not occurred since the last time the flag is cleared. 1 End of conversion of injected channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
29 JECH	End of injected chain conversion interrupt flag This bit indicates the end of conversion for an injected chain. 0 End of conversion of injected chain has not occurred since the last time the flag is cleared. 1 End of conversion of injected chain has occurred or the flag has not been cleared since the last end of conversion occurrence.
30 NEOC	End of normal channel conversion interrupt flag This bit indicates the end of conversion for a normal channel. 0 End of conversion of normal channel has not occurred since the last time the flag is cleared. 1 End of conversion of normal channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
31 NECH	End of normal chain conversion interrupt flag This bit indicates the end of conversion for a normal chain. 0 End of conversion of normal chain has not occurred since the last time the flag is cleared. 1 End of conversion of normal chain has occurred or the flag has not been cleared since the last end of conversion occurrence.

40.5.1.4 Internal Channel Interrupt Pending Registers 0–2 (ICIPR0–2)

The interrupt channel register to channel association is described in [Table 539](#).

Figure 472. Internal Channel Interrupt Pending Registers (ICIPR0–2)

Offset ICIPR0–2: 0x014–0x01C

Access: User Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 538. ICIPR0–2 field descriptions

Field	Description
0–31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 539. ICIPR0–2 Registers to channel association

Register	Register bits 31:0
ICIPR0	EOC_CH[31:0]
ICIPR1	EOC_CH[63:32]
ICIPR2	EOC_CH[95:64]

Note: Parameter *num_intch* vector has positional inference for registers ICIPR0–2, such that:

- *num_intch*(*i*) (where *i* = 0 to 31) corresponds to ICIPR0 bit [0 to 31],
- *num_intch*(*i*) (where *i* = 32 to 63) corresponds to ICIPR1 bit [0 to 31],
- and
- *num_intch*(*i*) (where *i* = 64 to 95) corresponds to ICIPR2 bit [0 to 31].

If any *num_intch*(*i*) value is '0', corresponding bit is not implemented and read access returns '0' on that bit location.

Thus,

- if *num_intch*[31:0] is all 0's, register ICIPR0 is not implemented.
- if *num_intch*[63:32] is all 0's, register ICIPR1 is not implemented.
- if *num_intch*[95:64] is all 0's, register ICIPR2 is not implemented.

Each is treated as reserved space; any access on this space generates a transfer error.

40.5.1.5 Interrupt Mask Register (IMR)

The description of the Interrupt Mask Register is given below.

Figure 473. Interrupt Mask Register (IMR)

Offset 0x020												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU ⁽¹⁾	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

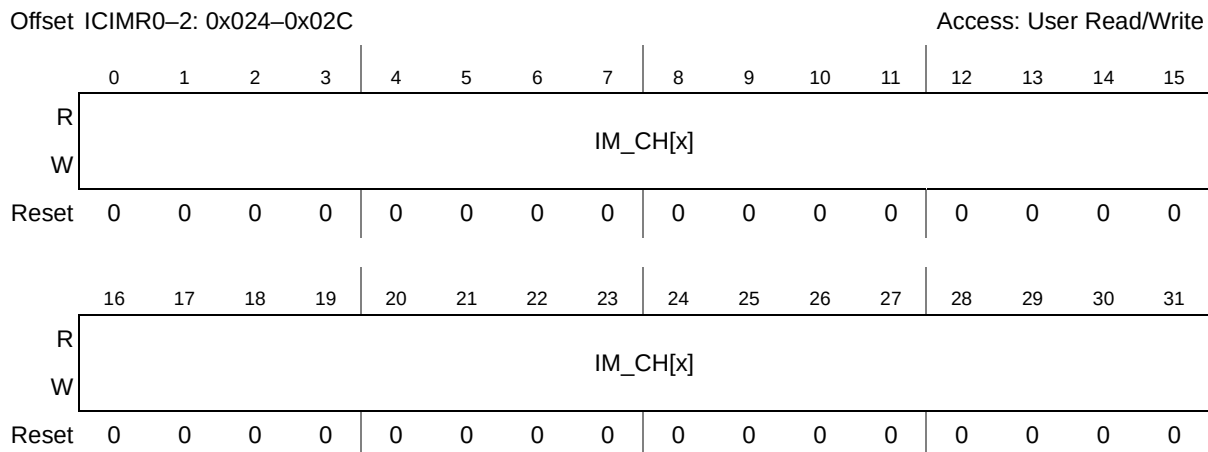
1. Only available with cross-triggering unit (CTU) feature.

Table 540. IMR field descriptions

Field	Description
0–26	Reserved Write of any value has no effect; read value is always 0.
27	Reserved
28 MSKJEOC	Mask bit for JEOC 0 JEOC interrupt is disabled. 1 JEOC interrupt is enabled.
29 MSKJECH	Mask bit for JECH 0 JECH interrupt is disabled. 1 JECH interrupt is enabled.
30 MSKNEOC	Mask bit for NEOC 0 NEOC interrupt is disabled. 1 NEOC interrupt is enabled.
31 MSKNECH	Mask bit for NECH 0 NECH interrupt is disabled. 1 NECH interrupt is enabled.

40.5.1.6 Internal Channel Interrupt Mask Registers 0–2 (ICIMR0–2)

The interrupt mask register to channel association is described in [Table 542](#).

Figure 474. Internal Channel Interrupt Mask Registers (ICIMR0–2)**Table 541. ICIMR0–2 field descriptions**

Field	Description
0–31 IM_CH[x]	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 542. Interrupt Mask Registers to Channel Association

Register	Register bits 31:0
ICIMR0	IM_CH[31:0]
ICIMR1	IM_CH[63:32]
ICIMR2	IM_CH[95:64]

Note: Parameter `num_intch` vector has positional inference for registers ICIMR0–2, such that

- `num_intch(i)` (where $i = 0$ to 31) corresponds to ICIMR0 bit [0 to 31],
- `num_intch(i)` (where $i = 32$ to 63) corresponds to ICIMR1 bit [0 to 31],

and

- `num_intch(i)` (where $i = 64$ to 95) corresponds to ICIMR2 bit [0 to 31].

If any of the `num_intch(i)` value is '0' corresponding bit is not implemented and read access would return '0' on that bit location.

Thus,

- if `num_intch[31:0]` is all 0's, register ICIMR0 is not implemented,
- if `num_intch[63:32]` is all 0's, register ICIMR1 is not implemented

and similarly

- if `num_intch[95:64]` is all 0's, register ICIMR2 is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.7 Watchdog Threshold Interrupt Status Register (WTISR)

This register gives the interrupt status information for the 16 possible upper/lower threshold limits monitored by upto 16 watchdog monitors which can be selected for each channel.

Figure 475. Watchdog Threshold Interrupt Status Register (WTISR)

Offset 0x030												Access: User Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 543. WTISR field descriptions

Field	Description
2x+1 WDGxH [x = 0–15]	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold as reported by WDG monitor 'x'. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
Bit 2x WDGxL [x = 0–15]	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold as reported by WDG monitor 'x'. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.

Note: In case num_watchdog generic parameter is zero, this register is not implemented.

40.5.1.8 Watchdog Threshold Interrupt Mask Register (WTIMR)

This register gives the interrupt mask information for the 16 possible upper/lower threshold limits which can be selected for each channel.

Figure 476. Watchdog Threshold Interrupt Mask Register (WTIMR)

Offset 0x034												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 544. WTIMR field descriptions

Field	Description
2x+1 MSKWDGxH [x = 0–15]	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDGxH is disabled. 1 Interrupt for WDGxH is enabled.
Bit 2x MSKWDGxL [x = 0–15]	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDGxL is disabled. 1 Interrupt for WDGxL is enabled.

Note: In case `num_watchdog` generic parameter is zero, this register is not implemented.

40.5.1.9 DMA Enable Register (DMAE)

The DMA Enable (DMAE) register sets up the DMA for use with the SARADC.

Figure 477. DMA Enable Register (DMAE)

Offset 0x040												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 545. DMAE register field descriptions

Field	Description
0–29	Reserved Write of any value has no effect; read value is always 0.
30 DCLR	DMA clear sequence enable 0 DMA request cleared by Acknowledge from DMA controller 1 DMA request cleared on read of data registers
31 DMAEN	DMA global enable 0 DMA feature disabled 1 DMA feature enabled

40.5.1.10 Internal Channel DMA Select Registers 0–2 (ICDSR0–2)

The DMA select registers to channel association is described in [Table 547](#).

Figure 478. Internal Channel DMA Select Registers (ICDSR0–2)

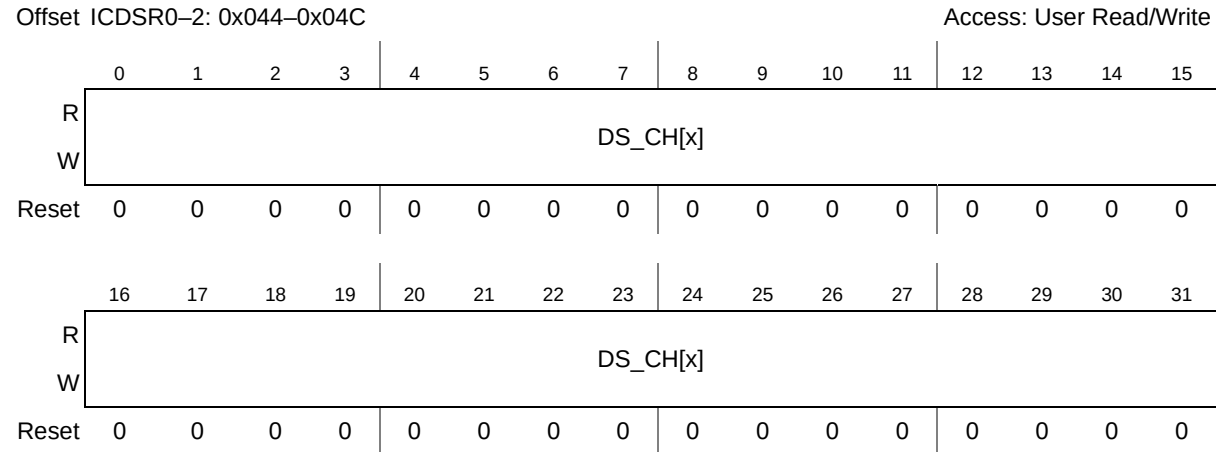


Table 546. ICDSR0–2 field descriptions

Field	Description
0–31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 547. Internal Channel DMA Select Registers to Channel Association

Register	Register bits 31:0
ICDSR0	DS_CH[31:0]
ICDSR1	DS_CH[63:32]
ICDSR2	DS_CH[95:64]

40.5.1.11 Watchdog Threshold Registers 0–3 (WTHRHLR0–3)

Figure 479. Watchdog Threshold Registers 0–3 (WTHRHLR0–3)

Offset WTHRHLR0–3: 0x060–0x06C												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	THRH											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	THRL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 548. WTHRHLR0–3 field descriptions

Field	Description
0–3	Reserved Write of any value has no effect; read value is always 0.
4–15 THRH	High threshold value for channel x
16–19	Reserved Write of any value has no effect; read value is always 0.
20–31 THRL	Low threshold value for channel x

Note: The number of WTHRHLR registers depends on num_watchdog generic parameter configuration.

In case num_watchdog is zero, these registers are not implemented.

The length of the threshold field depends on the generic resolution.

40.5.1.12 Conversion Timing Registers 0–3 (CTR0–3)

Figure 480. Conversion Timing Registers 0–3 (CTR0–3)

Offset CTR0–3: 0x094–0x0A0 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PRECHG				INPSAMP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 549. CTR0–3 field descriptions

Field	Description
0 CRES	Conversion Resolution select This bit selects the conversion resolution for the conversion. 0: 12-bit resolution 1: 10-bit resolution
1–19	Reserved Write of any value has no effect; read value is always 0.
20–23 PRECHG	Precharging phase duration This bitfield defines the precharging phase duration to be applied when this phase is enabled by PCE bit of channel data register. This duration is calculated as (PRECHG X 1/Frequency of SARADC clock). Please refer to device datasheet for the minimum duration required for precharging phase.
24–31 INPSAMP	Sampling phase duration This bitfield defines the sampling phase duration to be applied. This duration is calculated as (INPSAMP X 1/Frequency of SARADC clock). Please refer to device datasheet for the minimum duration required for sampling phase.

40.5.1.13 Internal Channel Normal Conversion Mask Registers 0–2 (ICNCMR0–2)

The normal conversion mask registers to channel association is described in [Table 551](#).

Figure 481. Internal Channel Normal Conversion Mask Registers (ICNCMR0–2)

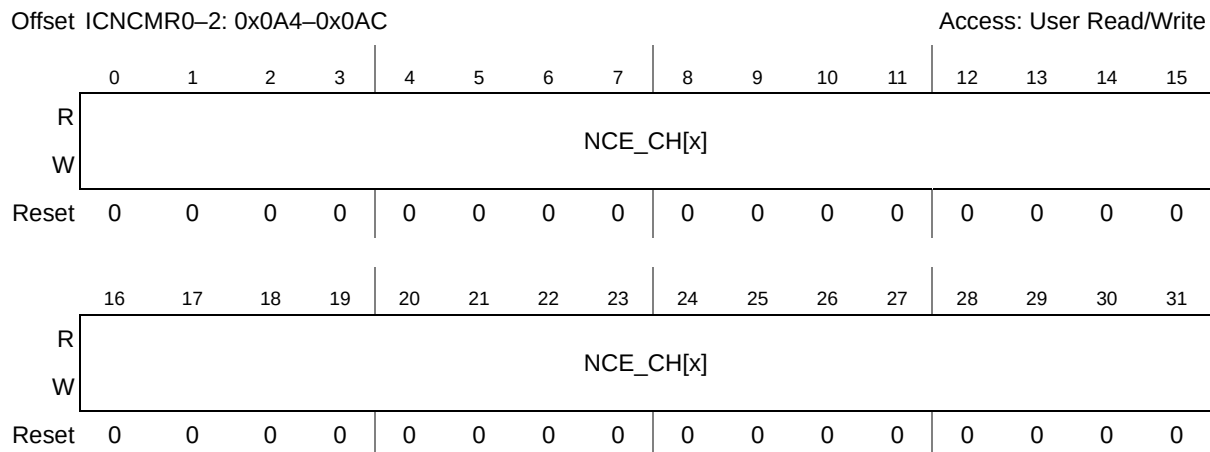


Table 550. ICNCMR0–2 field descriptions

Field	Description
0–31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 551. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 31:0
ICNCMR0	NCE_CH[31:0]
ICNCMR1	NCE_CH[63:32]
ICNCMR2	NCE_CH[95:64]

Note: Parameter *num_intch* vector has positional inference for registers ICNCMR0–2, such that

- *num_intch(i)* (where *i* = 0 to 31) corresponds to ICNCMR0 bit [0 to 31],
- *num_intch(i)* (where *i* = 32 to 63) corresponds to ICNCMR1 bit [0 to 31]

and

- *num_intch(i)* (where *i* = 64 to 95) corresponds to ICNCMR2 bit [0 to 31].

If any of the *num_intch(i)* value is '0' corresponding bit is not implemented and read access would return '0' on that bit location.

Thus

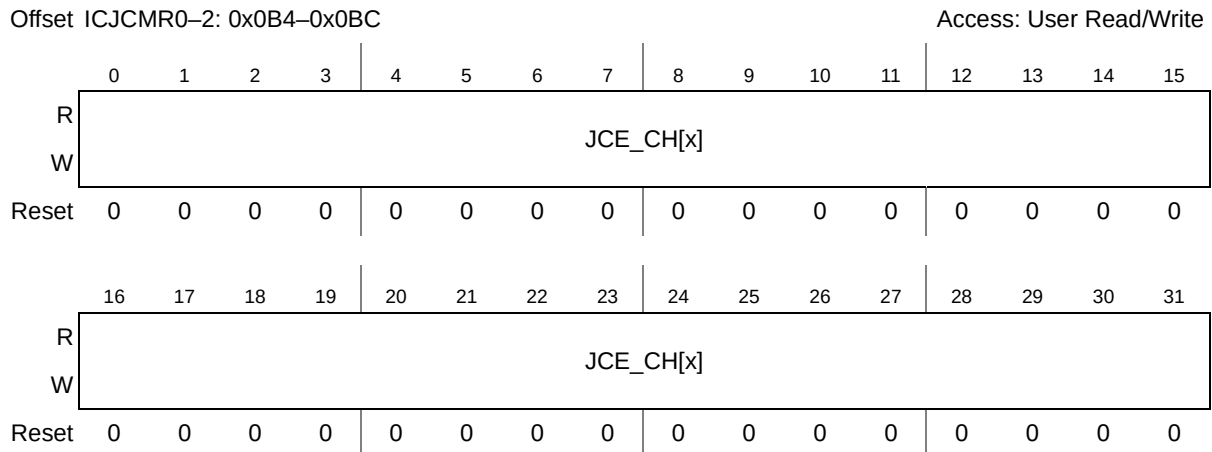
- if *num_intch*[31:0] is all 0's, register ICNCMR0 is not implemented
- if *num_intch*[63:32] is all 0's, register ICNCMR1 is not implemented

and similarly

- if *num_intch*[95:64] is all 0's, register ICNCMR2 is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.14 Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–2)

The injected conversion mask registers to channel association is described in [Table 553](#).

Figure 482. Internal Channel Injected Conversion Mask Registers (ICJCMR0–2)**Table 552. ICJCMR0–2 field descriptions**

Field	Description
0–31 JCE_CH[x]	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 553. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 31:0
ICJCMR0	JCE_CH[31:0]
ICJCMR1	JCE_CH[63:32]
ICJCMR2	JCE_CH[95:64]

Note: Parameter *num_intch* vector has positional inference for registers ICJCMR0–2, such that

- *num_intch(i)* (where *i* = 0 to 31) corresponds to ICJCMR0 bit [0 to 31],
- *num_intch(i)* (where *i* = 32 to 63) corresponds to ICJCMR1 bit [0 to 31]

and

- *num_intch(i)* (where *i* = 64 to 95) corresponds to ICJCMR2 bit [0 to 31].

If any of the *num_intch(i)* value is '0' corresponding bit is not implemented and read access would return '0' on that bit location.

Thus,

- if *num_intch*[31:0] is all 0's, register ICJCMR0 is not implemented,
- if *num_intch*[63:32] is all 0's, register ICJCMR1 is not implemented

and similarly

- if *num_intch*[95:64] is all 0's, register ICJCMR2 is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.15 Power Down Exit Delay Register (PDEDR)

Figure 483. Power Down Exit Delay Register (PDEDR)

Offset PDEDR: 0x0C8												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PDED							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 554. PDEDR field descriptions

Field	Description
0–23	Reserved Write of any value has no effect; read value is always 0.
24–31 PDED	PDED: Power down exit delay duration. Defines the delay between the power-down bit reset and the starting of conversion. The power-down delay is calculated as $[PDED \times 1/(SARADC \text{ clock frequency})]$.

40.5.1.16 Internal Channel Data Registers 0–95 (ICDR0–95)

The conversion results for each channel is loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid and some control information to select the required reference voltage, timing parameter selection for each channel.

Figure 484. Internal Channel Data Registers (ICDR0–95)

Offset ICDR0–95: 0x100–0x27C

Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW		RESULT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CDATA											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 555. ICDR0–95 field descriptions

Field	Description
0	Reserved Write of any value has no effect; read value is always 0.
1 REFSEL	Reference selection This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (e.g., : 5 V) 1 Selects the alternate reference (e.g., : 2 V)
2–3	Reserved Write of any value has no effect; read value is always zero.
4 PCE	<i>Precharge Enable</i> This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled
5	Reserved. Write of any value has no effect; read value is always zero.
6–7 CTSEL	This bitfield selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected
8–11	Reserved Write of any value has no effect; read value is always 0.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.

Table 555. ICDR0–95 field descriptions (continued)

Field	Description
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14–15 RESULT[1:0]	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16–19	Reserved Write of any value has no effect; read value is always 0.
20–31 CDATA	Channel converted data

Note: The length of the CDATA field depends on the generic resolution.

As each of ICDRx corresponds to an internal channel inferred through generic num_intch. If any of the num_intch(i) is '0', the corresponding ICDRx is not implemented and any access to it generates transfer error. This register should be accessed 32-bit R/W.

The REFSEL bit implementation is dependent on parameter vector altref_exists_ch[95:0].

40.5.1.17 Watchdog Threshold Registers 4–15 (WTHRHLR4–15)

Figure 485. Watchdog Threshold Registers 4–15 (WTHRHLR4–15)

Offset WTHRHLR4–15: 0x280–0x28C																Access: User Read/Write							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15							
R	0	0	0	0	THRH																		
W																							
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1							

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	0	0	0	0	THRL															
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Table 556. WTHRHLR4–15 field descriptions

Field	Description
0–3	Reserved Write of any value has no effect; read value is always 0.
4–15 THRH	High threshold value for channel x
16–19	Reserved Write of any value has no effect; read value is always 0.
20–31 THRL	Low threshold value for channel x

Note: The number of WTHRHLR registers depends on num_watchdog generic parameter configuration.

In case num_watchdog is zero, these registers are not implemented.

The length of the threshold field depends on the generic resolution.

40.5.1.18 Internal Channel Watchdog Select Registers (ICWSELR0–11)

These registers contain WSEL_CHx[3:0] bitfields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described in [Table 558](#).

Figure 486. Internal Channel Watchdog Select Registers (ICWSELR0–11)

Offset ICWSELR0–11: 0x2B0–0x2DC												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WSEL_CH7				WSEL_CH6				WSEL_CH5				WSEL_CH4			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WSEL_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 557. ICWSELR0–11 field descriptions

Field	Description
4*x+3 : 4*x WSEL_CH[x]	Watchdog select for channel x 0000: THRHLR0 register is selected 0001: THRHLR1 register is selected 0010: THRHLR2 register is selected 1110: THRHLR14 register is selected 1111: THRHLR15 register is selected

Table 558. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
ICWSELRx (x = 0 to 11)	WSEL_CH[(8*x)+7] .. WSEL_CH[8*x]

Note: Each such nibble (i.e., WSEL_CHx) has positional correspondence to num_intch parameter (each channel has a 1:1 nibble correspondence), so each word ICWSELRx corresponds to eight internal channels, thus requiring total twelve 32-bit registers to cover all 96 internal channels. If the value of any num_intch(i) is '0', the corresponding WSEL_CHx nibble is not implemented.

If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

40.5.1.19 Internal Channel Watchdog Enable Registers 0–2 (ICWENR0–2)

The Watchdog enable registers to channel association is described in [Table 560](#).

Figure 487. Internal Channel Watchdog Enable Registers (ICWENR0–2)

Offset ICWENR0–2: 0x2E0–0x2E8												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WEN_CH[x]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WEN_CH[x]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 559. ICWENR0–2 field descriptions

Field	Description
0–31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 560. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
ICWER0	WEN_CH[31:0]
ICWER1	WEN_CH[63:32]
ICWER2	WEN_CH[95:64]

Note: Parameter *num_intch* vector has positional inference for registers ICWER0–2, such that:

- *num_intch(i)* (where *i* = 0:31) corresponds to ICWER0 bit [0:31]
- *num_intch(i)* (where *i* = 32:63) corresponds to ICWER1 bit [0:31]

and

- *num_intch(i)* (where *i* = 64:95) corresponds to ICWER2 bit [0:31]

If any *num_intch(i)* value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus,

- if *num_intch*[31:0] is all 0's, register ICWER0 is not implemented,
- if *num_intch*[63:32] is all 0's, register ICWER1 is not implemented

and similarly,

- if *num_intch*[95:64] is all 0's, register ICWER2 is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.20 Internal Channel Analog Watchdog Out of Range Registers 0–2 (ICAWORR0–2)

The analog watchdog registers to channel association is described in [Table 562](#).

Figure 488. Internal Channel Analog Watchdog Out of Range Registers (ICAWORR0–2)

Offset ICAWORR0–2: 0x2F0–0x2F8												Access: User Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 561. ICAWORR0–2 field descriptions

Field	Description
0–31 AWOR_CH[x]	AWOR_CH[x]: Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Note: In case either num_watchdog is zero, these registers are not implemented.

Table 562. Internal Channel Analog Watchdog Registers to Channel Association

Register	Register bits 31:0
ICAWORR0	AWOR_CH[31:0]
ICAWORR1	AWOR_CH[63:32]
ICAWORR2	AWOR_CH[95:64]

Note: Parameter num_intch vector has positional inference for registers ICAWORR0–2, such that

- num_intch(i) (where i = 0:31) corresponds to ICAWORR0 bit [0:31]
- num_intch(i) (where i = 32:63) corresponds to ICAWORR1 bit [0:31]

and

- num_intch(i) (where i = 64:95) corresponds to ICAWORR2 bit [0:31]

If any of the num_intch(i) value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus,

- if num_intch[31:0] is all 0's, register ICAWORR0 is not implemented
- if num_intch[63:32] is all 0's, register ICAWORR1 is not implemented

and similarly,

- if num_intch[95:64] is all 0's, register ICAWORR2 is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.21 Test Channel Interrupt Pending Register (TCIPR)

The interrupt channel register to channel association is described in [Table 539](#).

Figure 489. Test Channel Interrupt Pending Register (TCIPR)

Offset TCIPR: 0x400												Access: User Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 563. TCIPR field descriptions

Field	Description
0–31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 564. TCIPR Register to Channel Association

Register	Register bits 31:0
TCIPR	EOC_CH[127:96]

Note: Parameter *num_testch* vector has positional inference for registers *TCIPR*, such that *num_testch(i)* (where *i* = 0:31) corresponds to *TCIPR* bit [0:31].

If any of the *num_testch(i)* value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if *num_testch* is all 0's, register *TCIPR* is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.22 Test Channel Interrupt Mask Registers (TCIMR)

The interrupt mask register to channel association is described in [Table 542](#).

Figure 490. Test Channel Interrupt Mask Registers (TCIMR)

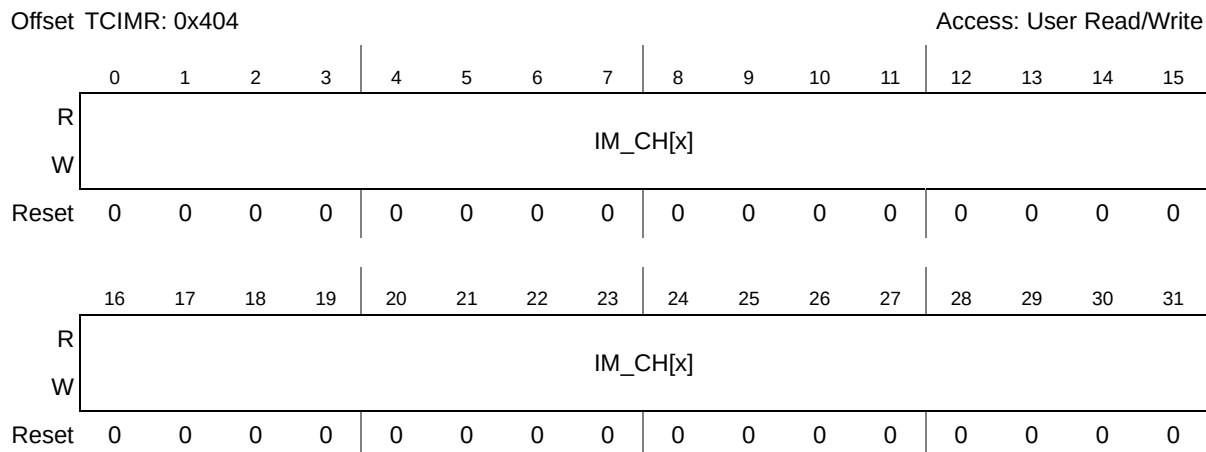


Table 565. TCIMR field descriptions

Field	Description
0–31 IM_CH[x]	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 566. Test Channel Interrupt Mask Register to Channel Association

Register	Register bits 31:0
TCIMR	IM_CH[127:96]

Note: *Parameter num_testch vector has positional inference for registers TCIMR, such that num_testch(i) (where i = 0:31) corresponds to TCIMR bit [0:31].*

If any of the num_testch(i) value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if the complete num_testch is all 0's, TCIMR is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.23 Test Channel DMA Select Register (TCDSR)

The DMA select registers to channel association is described in [Table 547](#).

Figure 491. Test Channel DMA Select Register (TCDSR)

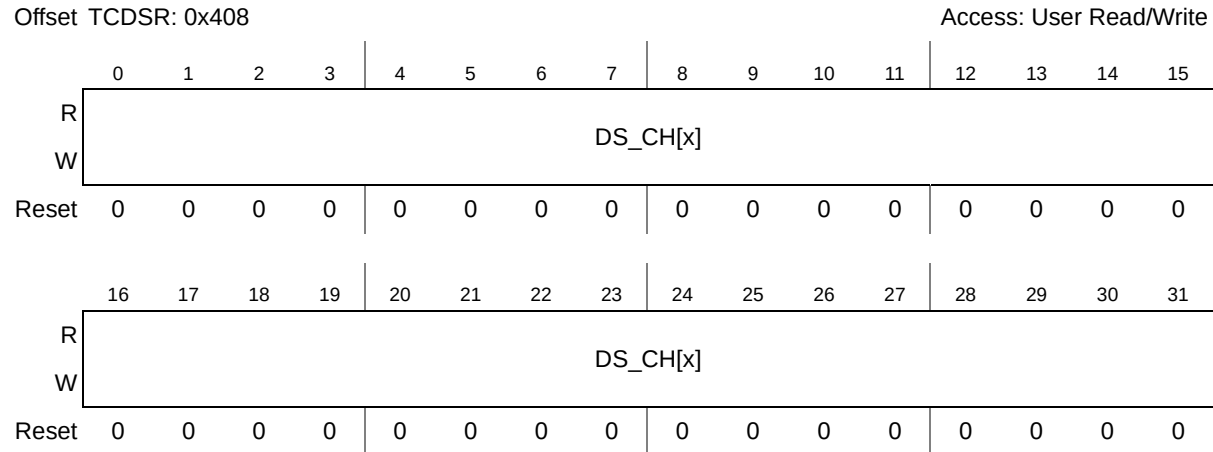


Table 567. TCDSR field descriptions

Field	Description
0–31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 568. Test Channel DMA Select Registers to Channel Association

Register	Register bits 31:0
TCDSR	DS_CH[127:96]

Note: *Parameter num_testch vector has positional inference for register TCDSR, such that num_testch(i) (where i = 0:31) corresponds to TCDSR bit [0:31].*

If any of the num_testch(i) value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if the complete num_testch is all 0's, TCDSR is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.24 Test Channel Normal Conversion Mask Register (TCNCMR)

The normal conversion mask register to channel association is described in [Table 551](#).

Figure 492. Test Channel Normal Conversion Mask Registers (TCNCMR)

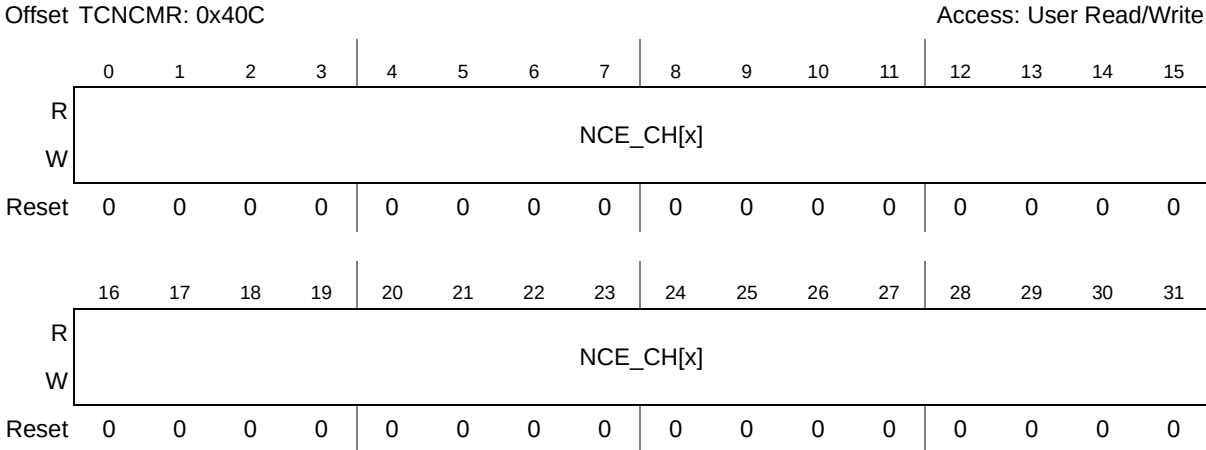


Table 569. TCNCMR field descriptions

Field	Description
0–31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 570. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register bits 31:0
TCNCMR	NCE_CH[127:96]

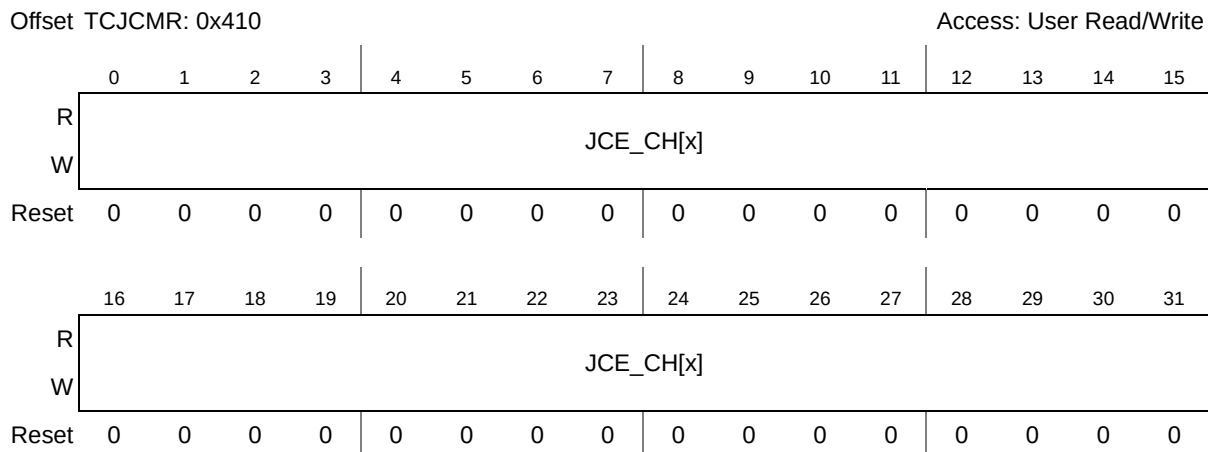
Note: *Parameter num_testch vector has positional inference for registers TCNCMR, such that num_testch(i) (where i = 0:31) corresponds to TCNCMR bit [0:31].*

If any of the num_testch(i) value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if the complete num_testch is all 0's, TCNCMR is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.25 Test Channel Injected Conversion Mask Register (TCJCMR)

The injected conversion mask register to channel association is described in [Table 572](#).

Figure 493. Test Channel Injected Conversion Mask Register (TCJCMR)**Table 571. TCJCMR field descriptions**

Field	Description
0–31 JCE_CH[x]	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 572. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register bits 31:0
TCJCMR	JCE_CH[127:96]

Note: Parameter *num_testch* vector has positional inference for registers, such that *num_testch(i)* (where *i* = 0:31) corresponds to TCJCMR bit [0:31].

If any of the *num_testch(i)* value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if the complete *num_testch* is all 0's, TCJCMR is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.26 Test Channel Watchdog Select Registers 0–3 (TCWSELR0–3)

This register contains WSEL_CHx[3:0] bitfields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select register to channel association is described in [Table 558](#).

Figure 494. Test Channels Watchdog Select Registers (TCWSELR)

Offset TCWSELR: 0x414–0x420																Access: User Read/Write			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	WSEL103				WSEL_CH102				WSEL_CH101				WSEL_CH100						
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	WSEL_CH99				WSEL_CH98				WSEL_CH97				WSEL_CH96						
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 573. TCWSELR field descriptions

Field	Description
4*x+3 : 4*x WSEL_CH[x]	Watchdog select for channel x 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected

Table 574. Test Channel Watchdog Select Registers to Channel Association

Register	Register bits 31:0
TCWSELR0	WSEL_CH[103] .. WSEL_CH[96]
TCWSELR1	WSEL_CH[111] .. WSEL_CH[104]
TCWSELR2	WSEL_CH[119] .. WSEL_CH[112]
TCWSELR3	WSEL_CH[127] .. WSEL_CH[120]

Note: Number of bits implemented for each WSEL_CHx depends on num_watchdog parameter. If num_watchdog > 8, the number of bits implemented will be 4 (i.e. WSEL_CHx[3:0]). If num_watchdog > 4 but <= 8, the number of bits implemented will be 3 (i.e. WSEL_CHx[3] will be reserved). Similarly, if num_watchdog > 1 but <= 4, the number of bits implemented will be 2 (i.e. WSEL_CHx[3:2] will be reserved). If num_watchdog = 1, only 1 bit is present. Each such nibble (i.e., WSEL_CHx) has positional correspondence to num_testch parameter (each channel has a 1:1 nibble correspondence), so each word TCWSELRx corresponds to eight test Channels, thus requiring total four 32-bit registers to cover all 32

test channels. If value of any `num_testch(i)` is '0', the corresponding `WSEL_CHx` nibble is not implemented.

If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

40.5.1.27 Test Channel Watchdog Enable Register (TCWENR)

The Watchdog enable registers to channel association is described in [Table 560](#).

Figure 495. Test Channel Watchdog Enable Register (TCWENR)

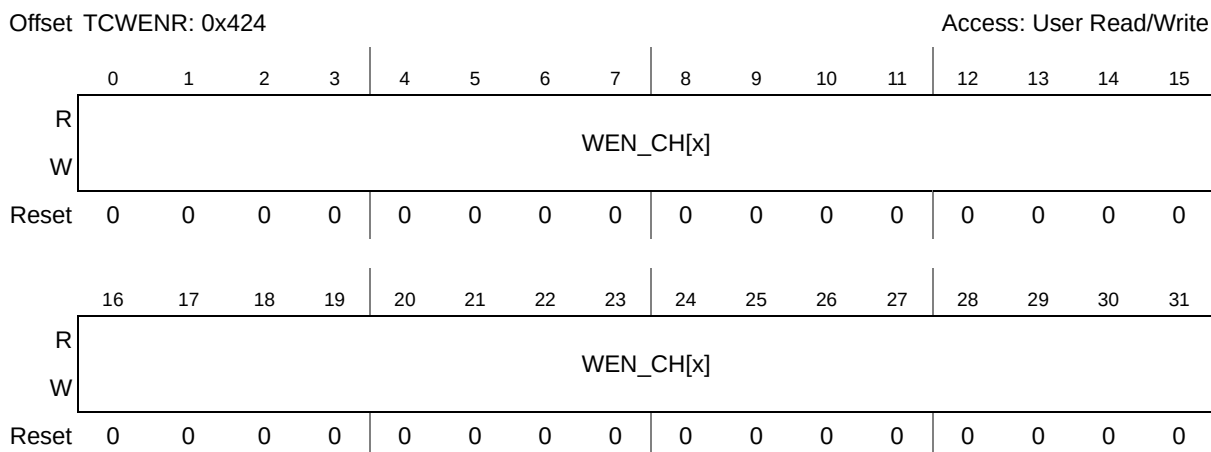


Table 575. TCWENR field descriptions

Field	Description
0–31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 576. Test Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
TCWENR	WEN_CH[127:96]

Note: Parameter `num_testch` vector has positional inference for registers `TCWENR`, such that `num_testch(i)` (where $i = 0:31$) corresponds to `TCWENR` bit [0:31].

If any of the `num_testch(i)` value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if the complete `num_testch` is all 0's, `TCWENR` is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.28 Test Channel Analog Watchdog Out of Range Registers (TCAWORR)

The analog watchdog registers to channel association is described in [Table 562](#).

Figure 496. Test Channel Analog Watchdog Out of Range Register (TCAWORR)

Offset TCAWORR: 0x0428												Access: User Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 577. TCAWORR field descriptions

Field	Description
0–31 AWOR_CH[x]	AWOR_CH[x]: Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Note: In case either of num_watchdog is zero, these registers are not implemented.

Table 578. Analog Watchdog Register to Channel Association

Register	Register bits 31:0
TCAWORR	AWOR_CH[127:96]

Note: Parameter num_testch vector has positional inference for registers TCAWORR, such that num_testch(i) (where i = 0:31) corresponds to TCAWORR bit [0:31].

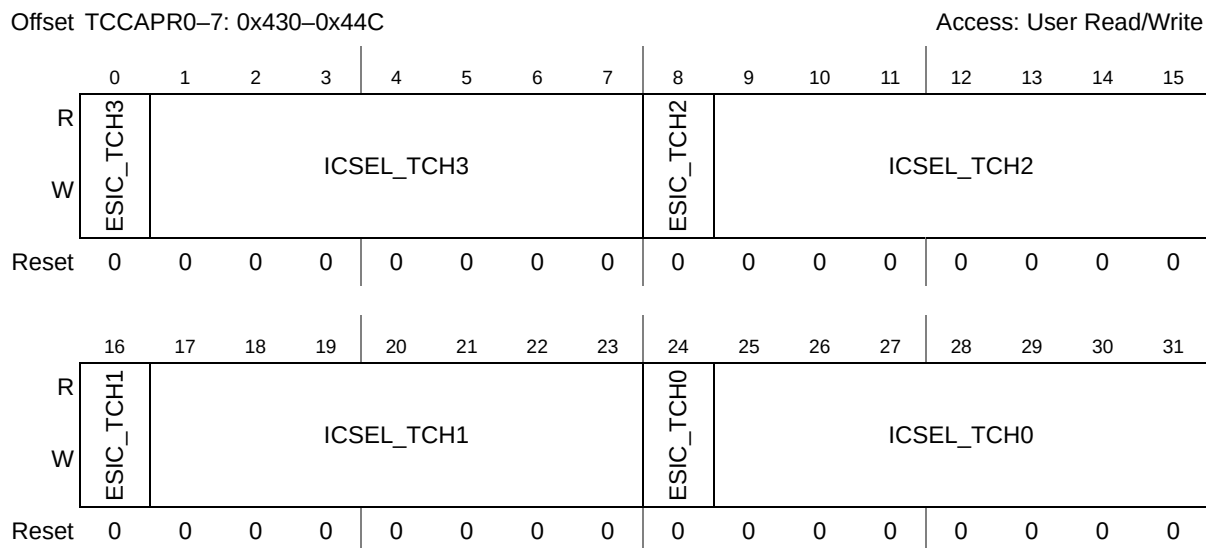
If any of the num_testch(i) value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if the complete num_testch is all 0's, TCAWORR is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.29 Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–7)

Each test channel can be shorted with any internal analog channel using the configuration programmed in these registers.

Test channel short registers to channel association is described in [Table 580](#).

Figure 497. Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–7)

For analog test channel assignment, refer to the device configuration or ADC configuration section.

Table 579. Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–7) field descriptions

Field (Little Endian MSB = bit 31)	Field (Big Endian MSB = bit 0)	Description
Bit 8x+7 ESIC_TCH[x]	Bit 24-8x ESIC_TCH[x]	Enable short with internal channel for test channel x 0 Test channel x cannot be shorted with internal channel 1 Test channel x can be shorted with internal channel
Bit 8x+6 : 8x ICSEL_TCH[x]	Bit 25-8x : 31-8x ICSEL_TCH[x]	Internal channel selection for short with test channel x

Table 580. Test Channel Short Registers to Channel Association

Register	Register bits 31:0 or 0:31 (arrangement of bit fields left to right)
TCCAPRx (x = 0 to 7)	ESIC_TCH[4*x+3], ICSEL_TCH[4*x+3], ESIC_TCH[4*x+2], ICSEL_TCH[4*x+2], ESIC_TCH[4*x+1], ICSEL_TCH[4*x+1], ESIC_TCH[4*x], ICSEL_TCH[4*x]

Note:

- 1) Parameter *num_testch* vector has positional inference for registers *TCCAPRx*, such that
 - *num_testch(i)* (where *i* = 0:3) corresponds to *TCCAPR0*,
 - *num_testch(i)* (where *i* = 4:7) corresponds to *TCCAPR1*,
 - and so on through
 - *num_testch(i)* (where *i* = 28:31) corresponds to *TCCAPR7*.

If all the `num_testch[3:0]` values are '0', the `TCCAPR0` word is not implemented, and so on, up to if all `num_testch[31:28]` values are '0' `TCCAPR7` word is not implemented and treated as reserved space. Any access on this space generates a transfer error.

2) An additional parameter `testch_short_intch` vector also has positional inference for registers `TCCAPRx`, such that

- `testch_short_intch(0)` corresponds to `TCCAPR0[7:0]`,
- `testch_short_intch(1)` corresponds to `TCCAPR0[15:8]`,
- `testch_short_intch(2)` corresponds to `TCCAPR0[23:16]`,
- and so on through
- `testch_short_intch(31)` corresponds to `TCCAPR7[31:24]`.

If `testch_short_intch(0)` value is '0' `TCCAPR0[7:0]` byte is not implemented and so on, up to if `testch_short_intch(31)` value is '0' `TCCAPR7[31:24]` byte is not implemented and treated as reserved space.

3) Parameter `num_testch` has higher precedence than `testch_map_enable`. For e.g. if `num_testch[3:0] = '0011'` and `testch_map_enable[3:0] = '1101'`, only `TCCAPR0[7:0]` byte will be implemented.

40.5.1.30 Test Channel Data Register 96–127 (TCDR96–127)

The conversion results for each channel is loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid and some control information to select the required reference voltage, timing parameter selection for each channel.

Figure 498. Test Channel Data Registers (TCDR96–127)

Offset TCDR96–127: 0x450–0x4CC

Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW		RESULT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CDATA											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 581. TCDR96–127 field descriptions

Field	Description
0	Reserved Write of any value has no effect; read value is always 0.
1 REFSEL	Reference selection This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (for e.g., : 5V) 1 Selects the alternate reference (for e.g., : 2V)
2–3	Reserved. Write of any value has no effect; read value is always zero.
4 PCE	<i>Precharge Enable</i> This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled
5	Reserved. Write of any value has no effect; read value is always zero.
6–7 CTSEL	This bitfield selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected
8–11	Reserved Write of any value has no effect; read value is always 0.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14–15 RESULT[1:0]	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16–19	Reserved Write of any value has no effect; read value is always 0.
20–31 CDATA	Channel converted data

Note: The length of the CDATA field depends on the generic resolution. As each of TCDRx corresponds to a test channel inferred through generic num_testch. If any of the num_testch(i) is '0', the corresponding TCDRx is not implemented and any access to it generates a transfer error. These registers are accessed 32-bit R/W.
The REFSEL bit implementation is dependent on parameter vector altref_exists_ch[127:96].

40.5.1.31 External Channel Decode Signals Delay Register (ECDSDR)

Figure 499. External Channel Decode Signals Delay Register (ECDSDR)

Offset ECDSDR: 0x500												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	DSD											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 582. ECDSDR field descriptions

Field	Description
0–19	Reserved Write of any value has no effect; read value is always 0.
20–31 DSD	Decode signals delay This bitfield defines the delay between the external decode signals and the start of the sampling phase. It is used to take into account of the settling time required for the external mux. The decode signal delay is calculated as (DSD X 1/Frequency of SARADC clock).

Note: If parameter num_extch is all 0's, the ECDSDR is not implemented and treated as reserved space. Any access on this space generates a transfer error.

40.5.1.32 External Channel Interrupt Pending Registers 0–3 (ECIPR0–3)

The interrupt channel register to channel association is described in [Table 539](#)

Figure 500. External Channel Interrupt Pending Register (ECIPR0–3)

Offset ECIPR0–3: 0x510–0x51C												Access: User Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 583. ECIPR0–3 register field descriptions

Field	Description
0–31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 584. ECIPR0–3 Registers to Channel Association

Register	Register bits 31:0
ECIPR0	EOC_CH[159:128]
ECIPR1	EOC_CH[191:160]
ECIPR2	EOC_CH[223:192]
ECIPR3	EOC_CH[255:224]

Note: Parameter `num_extch` vector has positional inference for registers ECIPR0–3, such that

- `num_extch(i)` (where $i = 0:31$) corresponds to ECIPR0 bit [0:31]
- `num_extch(i)` (where $i = 32:63$) corresponds to ECIPR1 bit [0:31]
- `num_extch(i)` (where $i = 64:95$) corresponds to ECIPR2 bit [0:31]

and

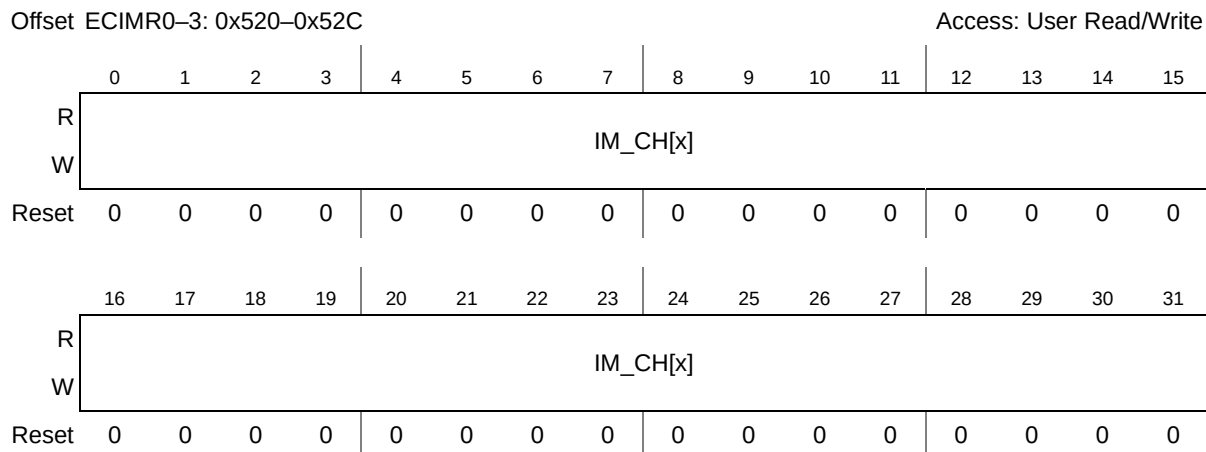
- `num_extch(i)` (where $i = 96:127$) corresponds to ECIPR3 bit [0:31].

If any of the `num_extch(i)` value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if any/all of `num_extch[31:0]/[63:32]/[95:64]/[127:96]` is all 0's, the access to corresponding register(s) i.e. ECIPR0/1/2/3 would generate a transfer error.

40.5.1.33 External Channel Interrupt Mask Registers 0–3 (ECIMR0–3)

The interrupt mask register to channel association is described in [Table 542](#).

Figure 501. External Channel Interrupt Mask Registers (ECIMR0–3)**Table 585. ECIMR0–3 field descriptions**

Field	Description
0–31 IM_CH[x]	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 586. External Channel Interrupt Mask Registers to Channel Association

Register	Register bits 31:0
ECIMR0	IM_CH[159:128]
ECIMR1	IM_CH[191:160]
ECIMR2	IM_CH[223:192]
ECIMR3	IM_CH[255:224]

Note: Parameter `num_extch` vector has positional inference for registers ECIMR0–3, such that

- `num_extch(i)` (where $i = 0:31$) corresponds to ECIMR0 bit [0:31]
- `num_extch(i)` (where $i = 32:63$) corresponds to ECIMR1 bit [0:31]
- `num_extch(i)` (where $i = 64:95$) corresponds to ECIMR2 bit [0:31]

and

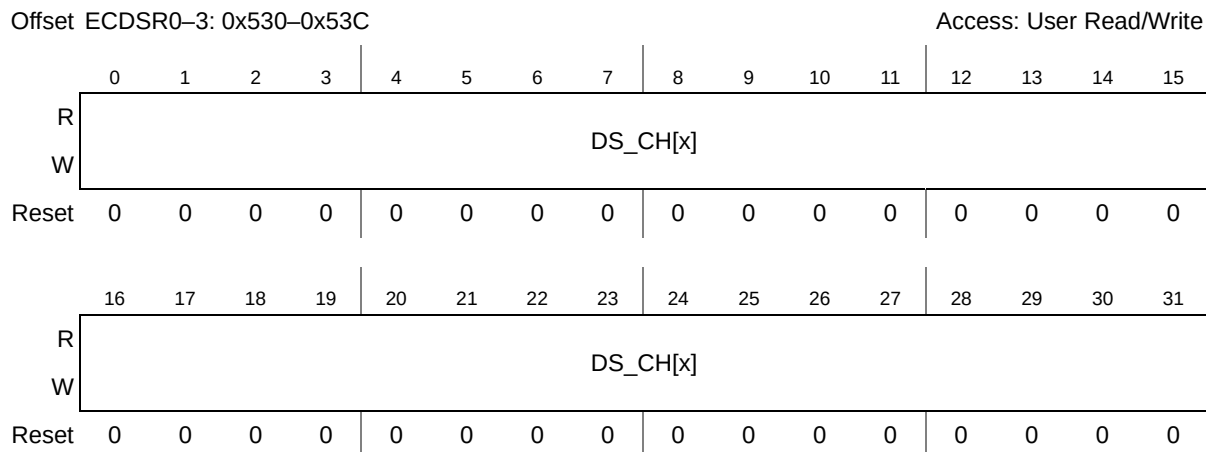
- `num_extch(i)` (where $i = 96:127$) corresponds to ECIMR3 bit [0:31].

If any of the `num_extch(i)` value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if any/all of `num_extch[31:0]/[63:32]/[95:64]/[127:96]` is all 0's, access to corresponding register(s) i.e. ECIMR0/1/2/3 would generate a transfer error.

40.5.1.34 External Channel DMA Select Registers 0–3 (ECDSR0–3)

The DMA select registers to channel association is described in [Table 547](#).

Figure 502. External Channel DMA Select Registers (ECDSR0–3)**Table 587. ECDSR0–3 field descriptions**

Field	Description
0–31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 588. DMA Select Registers to Channel Association

Register	Register bits 31:0
ECDSR0	DS_CH[159:128]
ECDSR1	DS_CH[191:160]
ECDSR2	DS_CH[223:192]
ECDSR3	DS_CH[255:224]

Note: Parameter `num_extch` vector has positional inference for registers ECDSR0–3, such that

- `num_extch(i)` (where $i = 0:31$) corresponds to ECDSR0 bit [0:31]
- `num_extch(i)` (where $i = 32:63$) corresponds to ECDSR1 bit [0:31]
- `num_extch(i)` (where $i = 64:95$) corresponds to ECDSR2 bit [0:31]

and

- `num_extch(i)` (where $i = 96:127$) corresponds to ECDSR3 bit [0:31].

If any of the `num_extch(i)` value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if any/all of `num_extch[31:0]/[63:32]/[95:64]/[127:96]` is all 0's, the access to corresponding register(s) i.e. ECDSR0/1/2/3 would generate a transfer error.

40.5.1.35 External Channel Normal Conversion Mask Registers 0–3 (ECNCMR0–3)

The normal conversion mask registers to channel association is described in [Table 551](#).

Figure 503. External Channel Normal Conversion Mask Registers (IECNCMR0–3)

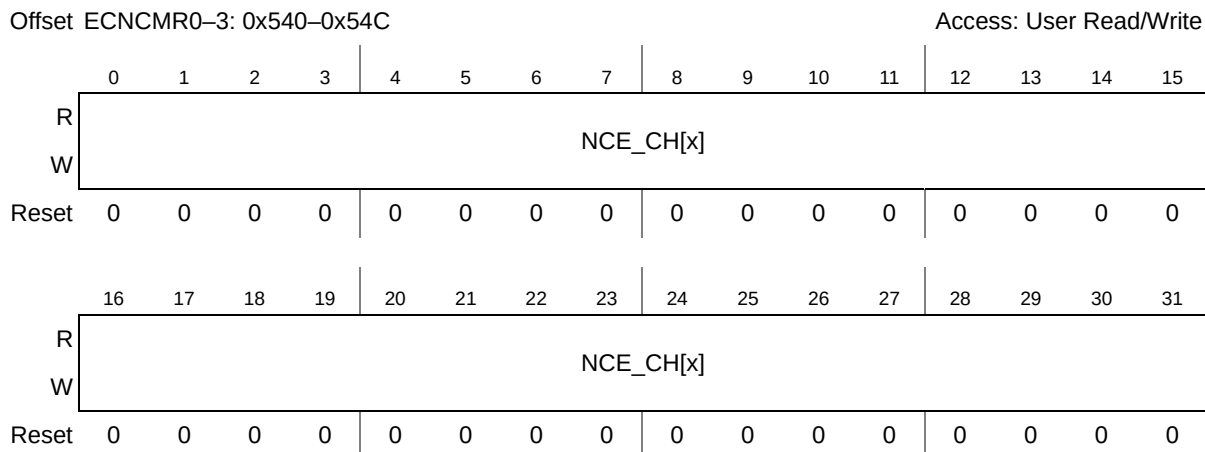


Table 589. ECNCMR0–3 field descriptions

Field	Description
0–31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 590. External Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 31:0
ECNCMR0	NCE_CH[159:128]
ECNCMR1	NCE_CH[191:160]
ECNCMR2	NCE_CH[223:192]
ECNCMR3	NCE_CH[255:224]

Note: Parameter *num_extch* vector has positional inference for registers ECNCMR0–3, such that

- *num_extch(i)* (where *i* = 0:31) corresponds to ECNCMR0 bit [0:31]
- *num_extch(i)* (where *i* = 32:63) corresponds to ECNCMR1 bit [0:31]
- *num_extch(i)* (where *i* = 64:95) corresponds to ECNCMR2 bit [0:31]

and

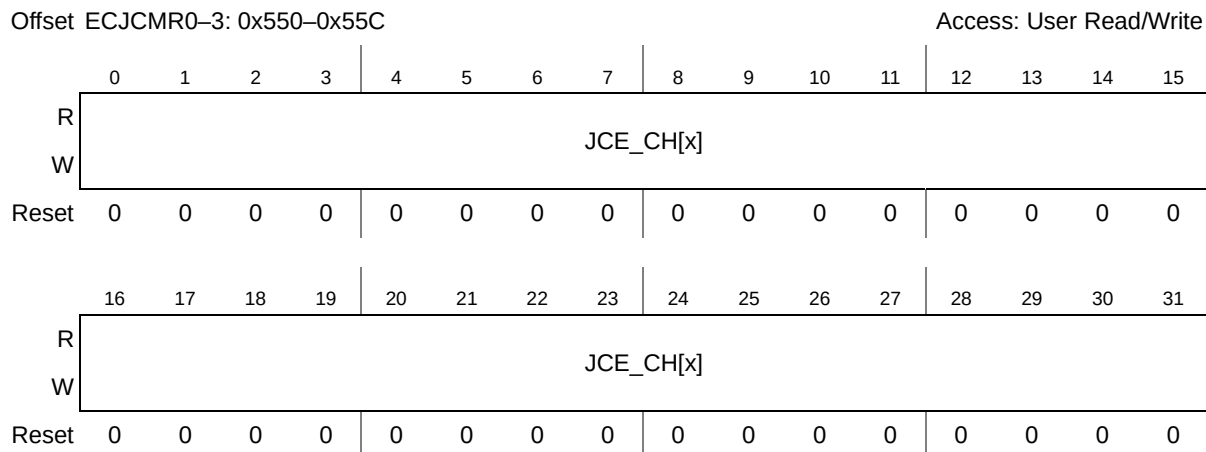
- *num_extch(i)* (where *i* = 96:127) corresponds to ECNCMR3 bit [0:31].

If any of the *num_extch(i)* value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if any/all of *num_extch*[31:0]/[63:32]/[95:64]/[127:96] is all 0's, the access to corresponding register(s) i.e. ECNCMR0/1/2/3 would generate a transfer error.

40.5.1.36 External Channel Injected Conversion Mask Registers 0–3 (ECJCMR0–3)

The injected conversion mask registers to channel association is described in [Table 592](#).

Figure 504. External Channel Injected Conversion Mask Registers (ECJCMR0–3)**Table 591. ECJCMR0–3 field descriptions**

Field	Description
0–31 JCE_CH[x]	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 592. External Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 31:0
ECJCMR0	JCE_CH[159:128]
ECJCMR1	JCE_CH[191:160]
ECJCMR2	JCE_CH[223:192]
ECJCMR3	JCE_CH[255:224]

Note: Parameter *num_extch* vector has positional inference for registers ECJCMR0–3, such that

- *num_extch(i)* (where *i* = 0:31) corresponds to ECJCMR0 bit [0:31]
- *num_extch(i)* (where *i* = 32:63) corresponds to ECJCMR1 bit [0:31]
- *num_extch(i)* (where *i* = 64:95) corresponds to ECJCMR2 bit [0:31]

and

- *num_extch(i)* (where *i* = 96:127) corresponds to ECJCMR3 bit [0:31].

If any of the *num_extch(i)* value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

Thus, if any/all of *num_extch*[31:0]/[63:32]/[95:64]/[127:96] are all 0's, the access to the corresponding register(s) i.e. ECJCMR0/1/2/3 would generate a transfer error.

40.5.1.37 External Channel Watchdog Select Registers (ECWSELR0–15)

These registers contain WSEL_CHx[3:0] bitfields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described in [Table 558](#).

Figure 505. External Channel Watchdog Select Registers (ECWSELR0–15)

Offset ECWSELR0–15: 0x0560–0x059C												Access: User Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WSEL_CH135				WSEL_CH134				WSEL_CH133				WSEL_CH132			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WSEL_CH131				WSEL_CH130				WSEL_CH129				WSEL_CH128			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 593. ECWSELR0–15 field descriptions

Field	Description
4*x+3 : 4*x WSEL_CH[x]	Watchdog select for channel x 0000: THRHLR0 register is selected 0001: THRHLR1 register is selected 0010: THRHLR2 register is selected 1110: THRHLR14 register is selected 1111: THRHLR15 register is selected

Table 594. External Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
ECWSELRx (x = 0 to 15)	WSEL_CH[(8*x)+135] .. WSEL_CH[(8*x)+128]

Note: Number of bits implemented for each WSEL_CHx depends on num_watchdog parameter:
 If num_watchdog > 8, the number of bits implemented will be 4 (i.e. WSEL_CHx[3:0]).
 If num_watchdog > 4 but <= 8, the number of bits implemented will be 3 (i.e. WSEL_CHx[3] will be reserved).
 Similarly, if num_watchdog > 1 but <= 4, the number of bits implemented will be 2 (i.e. WSEL_CHx[3:2] will be reserved).
 If num_watchdog = 1, then only 1 bit is present.
 Each such nibble (i.e. WSEL_CHx) has positional correspondence to num_extch parameter (each channel has a 1:1 nibble correspondence), so each word ECWSELRx corresponds to

eight external channels, thus requiring a total sixteen 32-bit registers to cover all 128 external channels. If value of any `num_extch(i)` is '0', the corresponding `WSEL_CHx` nibble is not implemented.

If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

40.5.1.38 External Channel Watchdog Enable Registers 0–3 (ECWENR0–3)

The Watchdog enable registers to channel association is described in [Table 560](#).

Figure 506. External Channel Watchdog Enable Registers (ECWENR0–3)

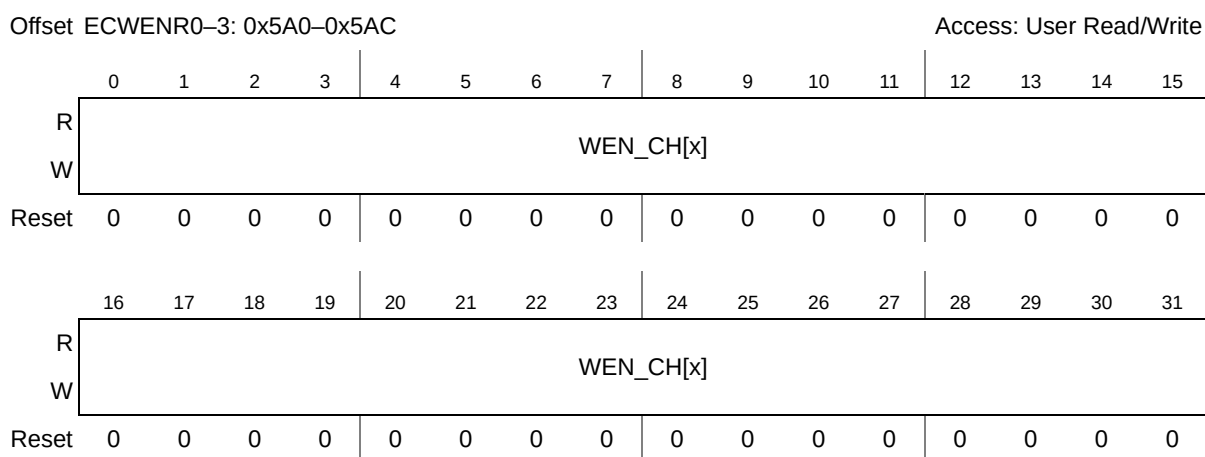


Table 595. ECWENR0–3 field descriptions

Field	Description
0–31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 596. External Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
ECWER0	WEN_CH[159:128]
ECWER1	WEN_CH[191:160]
ECWER2	WEN_CH[223:192]
ECWER3	WEN_CH[255:224]

Note: Parameter `num_extch` vector has positional inference for registers ECWER0–3, such that

- `num_extch(i)` (where $i = 0:31$) corresponds to ECWER0 bit [0:31]
- `num_extch(i)` (where $i = 32:63$) corresponds to ECWER1 bit [0:31]
- `num_extch(i)` (where $i = 64:95$) corresponds to ECWER2 bit [0:31],

and

– num_extch(i) (where $i = 96:127$) corresponds to ECWER3 bit [0:31].

If any num_extch(i) value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

If any/all of num_extch[31:0]/[63:32]/[95:64]/[127:96] are '0', the access to corresponding register(s) i.e. ECWER0/1/2/3 would generate a transfer error.

40.5.1.39 External Channel Analog Watchdog Out of Range Registers 0–3 (ECAWORR0–3)

The analog watchdog registers to channel association is described in [Table 562](#).

Figure 507. External Channel Analog Watchdog Out of Range Registers (ECAWORR0–3)

Offset	ECAWORR0–3: 0x05B0–0x05BC												Access: User Read Only			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 597. ECAWORR0–3 field descriptions

Field	Description
0–31 AWOR_CH[x]	AWOR_CH[x]: Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Note: In case either num_watchdog is zero, these registers are not implemented.

Table 598. External Channel Analog Watchdog Registers to Channel Association

Register	Register bits 31:0
ECAWORR0	AWOR_CH[159:128]
ECAWORR1	AWOR_CH[191:160]
ECAWORR2	AWOR_CH[223:192]
ECAWORR3	AWOR_CH[255:224]

Note: Parameter `num_extch` vector has positional inference for registers `ECAWORR0–3`, such that

- `num_extch(i)` (where $i = 0:31$) corresponds to `ECAWORR0` bit `[0:31]`
- `num_extch(i)` (where $i = 32:63$) corresponds to `ECAWORR1` bit `[0:31]`
- `num_extch(i)` (where $i = 64:95$) corresponds to `ECAWOERR2` bit `[0:31]`

and

- `num_extch(i)` (where $i = 96:127$) corresponds to `ECAWORR3` bit `[0:31]`.

If any `num_extch(i)` value is '0', the corresponding bit is not implemented and read access would return '0' on that bit location.

If any/all of `num_extch[31:0]/[63:32]/[95:64]/[127:96]` are '0', access to corresponding register(s) i.e. `ECAWORR0/1/2/3` would generate a transfer error.

40.5.1.40 External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–3)

Each set of eight external channels can be mapped with any internal analog channel using the configuration programmed in these registers.

Figure 508. External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–3)

Offset ECMICR0–3: 0x5C0–0x05CC																Access: User Read/Write							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15							
R	0	ICSEL_ECH152_159							0	ICSEL_ECH144_151													
W																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	0	ICSEL_ECH136_143							0	ICSEL_ECH128_135										
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Table 599. ECMICR0–3 field descriptions

Field	Description
Bit $8x+7$	Reserved Write of any value has no effect; read value is always 0.
Bit $8x+6 : 8x$ ICSEL_ECH	Internal channel selection for external channels $(128 + 8x)$ to $(135+8*x)$ where $x = 0,1,2..16$

Table 600. Test Channel Short Registers to Channel Association

Register	Register bits 31:0
ECMICRx ($x = 0$ to 3)	ICSEL_ECH[$128 + (32*x+31)$] _{128 + (32*x+24)} ICSEL_ECH[$128 + (32*x + 23)$] _{128 + (32*x+16)} , ICSEL_ECH[$128 + (32*x+15)$] _{128 + (32*x+8)} ICSEL_ECH[$128 + (32*x + 7)$] _{128 + 32*x} ,

Note: Parameter `num_extch` vector has positional inference for registers `ECMICR0–3`, such that

- `num_extch(i)` (where $i = 0:31$) corresponds to `ECMICR0`
- `num_extch(i)` (where $i = 32:63$) corresponds to `ECMICR1`
- `num_extch(i)` (where $i = 64:95$) corresponds to `ECMICR2`, and
- `num_extch(i)` (where $i = 96:127$) corresponds to `ECMICR3`.

If any/all of `num_extch[31:0]/[63:32]/[95:64]/[127:96]`, these parameter slices are all '0's, the access to corresponding register(s) i.e. `ECMICR0/1/2/3` would generate a transfer error.

40.5.1.41 External Channel Data Registers 128–255 (ECDR128–255)

The conversion results for each channel is loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid and some control information to select the required reference voltage, timing parameter selection for each channel.

Figure 509. External Channel Data Registers (ECDR128–255)

Offset ECDR128–255: 0x5D0–0x7CC																Access: User Read/Write													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15													
R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT														
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31													
R	0	0	0	0	CDATA																								
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													

Table 601. ECDR128–255 field descriptions

Field	Description
0	Reserved Write of any value has no effect; read value is always 0.
1 REFSEL	Reference selection This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (for e.g., : 5 V) 1 Selects the alternate reference (for e.g., : 2 V)
2–3	Reserved. Write of any value has no effect; read value is always zero.
4 PCE	<i>Precharge Enable</i> This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled

Table 601. ECDR128–255 field descriptions (continued)

Field	Description
5	Reserved. Write of any value has no effect; read value is always zero.
6–7 CTSEL	This bitfield selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected
8–11	Reserved Write of any value has no effect; read value is always 0.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14–15 RESULT[1:0]	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16–19	Reserved Write of any value has no effect; read value is always 0.
20–31 CDATA	Channel converted data

Note: The length of the CDATA field depends on the generic resolution. As each of ECDRx corresponds to an external channel inferred through generic num_extch. If the value of any num_extch(i) is '0', the corresponding ECDRx is not implemented. Any access on this space generates transfer error. This register should be accessed 32-bit R/W.

The REFSEL bit implementation is dependent on parameter vector altref_exists_ch[255:128].

40.6 Start of conversion pulse delay

The following table lists delays between various trigger point (initiation request for conversion as registered at system clock) and actual issue of start pulse (adc_start) to the ADC hardmacrocell. Here each possible trigger source is described with the number of

Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface

clock cycles required for different operations until trigger of `adc_start`. This table donot take into account any software delay involved in programming or polling of any bit in order to measure conversion timing.

Table 602. Start of conversion pulse delay table

Trigger source	Register control (<code>ipg_clk_s</code>)	Inter domain synchronization (<code>ipg_clk</code>)	ADC state machine (<code>ipg_clk_adc</code>)	Inter domain synchronization (<code>ipg_clk_adc</code>)	Comments
Normal conversion triggered by software (setting of NSTART of MCR)	1	—	2	$1-2^2$	Cumulative Delay from <code>wen(byte_en)</code> assertion to actual issue of <code>adc_start</code> pulse = 3-4 <code>ipg_clk_adc</code> cycles + 1 <code>ipg_clk</code> cycle.
Normal conversion triggered by external trigger (HW trigger through <code>ipp_ind_start_trg</code> port)	—	—	2	$1-2^2$	Cumulative delay from the triggering edge (rising or falling) to the actual issue of <code>adc_start</code> pulse = 3-4 <code>ipg_clk_adc</code> cycles.
Normal conversion within a chain	—	—	0 (from EOC of previous conversion) 1 (from <code>last_evaluation_cyc</code> of previous conversion)	—	Delay from the EOC of last conversion, The start of next conversion is issued exactly on the same cycle when EOC is expected for previous conversion, start pulse in issued looking at <code>last_evaluation_cyc_i</code> which is received one cycle prior to EOC.
Injected conversion triggered by software (setting of JSTART bit of MCR)	1	—	3	$1-2^2$	Cumulative delay from <code>wen(byte_en)</code> assertion. Here delay for first start pulse of the injected chain execution when FSM was originally in IDLE state is considered = 4-5 <code>ipg_clk_adc</code> cycles + 1 <code>ipg_clk</code> cycle.
Injected conversion triggered by external Trigger (HW trigger through <code>ipp_ind_injection_trg</code> port)	—	—	3	$1-2^2$	Cumulative delay from triggering edge (rising or falling) to actual issue of <code>adc_start</code> = 4-5 <code>ipg_clk_adc</code> cycles.

Table 602. Start of conversion pulse delay table (continued)

Trigger source	Register control (ipg_clk_s)	Inter domain synchronization (ipg_clk)	ADC state machine (ipg_clk_adc)	Inter domain synchronization (ipg_clk_adc)	Comments
Injected conversion within a chain (delay between consecutive conversion of injected chain)	—	—	0 (from EOC of previous conversion) 1 (from last_evaluation_cyc of previous conversion)	—	Delay from EOC of last injected channel and adc_start pulse of next injected channel. The start of next conversion is issued exactly on the same cycle when EOC is expected for previous conversion, start pulse is issued looking at last_evaluation_cyc_i which is received one cycle prior to EOC
Injected conversion over normal conversion (software injected conversion chain)	1	1–2 ²	3–5 ¹	1–2 ²	Cumulative delay from wen(byte_en) assertion to actual trigger of first injected channel, when the FSM was in conversion cycle of previous normal conversion = 4–7 ipg_clk_adc cycles + 2–3 ipg_clk cycles.
Injection of conversion over normal conversion in JTRGSEQ mode	—	—	3	1–2 ²	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 ipg_clk_adc cycles.
Abort of a Normal/Injected Channel conversion during Normal Chain/Injected Chain execution (setting of MCR[ABORT] bit)	1	1–2 ²	3–5 ⁽¹⁾	1–2 ⁽²⁾	Delay from writing of ABORT bit (wen/byte_en assertion) to actual issue of adc_start pulse for the next channel conversion = 4–7 ipg_clk_adc cycles + 2–3 ipg_clk cycles.

1. The above delay values are calculated over a range of system and ADC clock frequencies. With ipg_clk(max) = 50 MHz to ipg_clk(min) = 40 MHz and ipg_clk_adc(max) = 16 MHz to ipg_clk_adc(min) = 14 MHz. L.H.S value corresponds to max frequencies and RHS corresponds to min frequencies.

2. The above variation is owing to inter domain synchronization.

41 Temperature Sensor

41.1 Introduction

SPC57EM80xx MCUs include an onboard temperature sensor that monitors device temperature and delivers one analog output signal and three digital output signals. The analog output consists of a voltage signal directly proportional to the internal junction temperature. The analog output is connected to an input channel of an ADC on the device. The internal junction temperature must be calculated by software based on the sampled temperature sensor voltage, sampled bandgap voltage, which comes from another module, and calibration parameter values stored in internal flash memory. The three digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions. The digital outputs signal when the junction temperature drifts below the low temperature threshold or above one of two high temperature thresholds. These signals notify the device to take action to appropriately adjust the device temperature in response to an out of specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

41.1.1 Signal Descriptions

This module has no external signals.

41.2 Functional description

The temperature sensor generates one analog output voltage, $V_{TSENS}(T)$, which is proportional to the absolute current junction temperature of the device, and three digital outputs that signal whether the junction temperature has reached either a pre-set low temperature threshold or one of two pre-set high temperature thresholds.

An onboard ADC module is used to convert the analog output voltage, as well as the bandgap voltage, into a digital representation. These values, along with parameter values stored in onboard flash memory, are used by software to calculate the device junction temperature.

41.2.1 Temperature threshold detection (digital output generation)

Temperature threshold trimming values are adjusted through calibration during factory test and stored in onboard flash memory. While enabled, if a new trimming value is loaded into the Temperature Sensor module, it may take up to 1 μ s for the digital outputs to be updated reflecting the new trimming condition.

When the temperature threshold detection feature is enabled, the temperature sensor monitors the internal junction temperature of the chip and asserts a signal if any of the three specified temperature thresholds are crossed.

- The low temperature digital output signals if the junction temperature falls below the low temperature threshold ($-40\text{ }^{\circ}\text{C}$).
- The high temperature digital output 1 signals if the junction temperature rises above the first high temperature threshold ($150\text{ }^{\circ}\text{C}$).
- The high temperature digital output 2 signals if the junction temperature rises above the second high temperature threshold ($165\text{ }^{\circ}\text{C}$).

The threshold detection circuit is calibrated in respect to the factory-test temperatures: the low temperature threshold is trimmed to match the cold insertion test temperature (nominally equal to $-40\text{ }^{\circ}\text{C}$) and the high temperature thresholds are trimmed so that one of them matches the hot insertion test temperature (nominally equal to $150\text{ }^{\circ}\text{C}$ or $165\text{ }^{\circ}\text{C}$). In other words, the thresholds are not trimmed to absolute values of temperature but to the specific temperatures for which the device is validated during factory test.

Hysteresis is applied whenever outputs transition to a logic level low in order to eliminate spurious toggling. Digital output behavior is illustrated in [Figure 510](#).

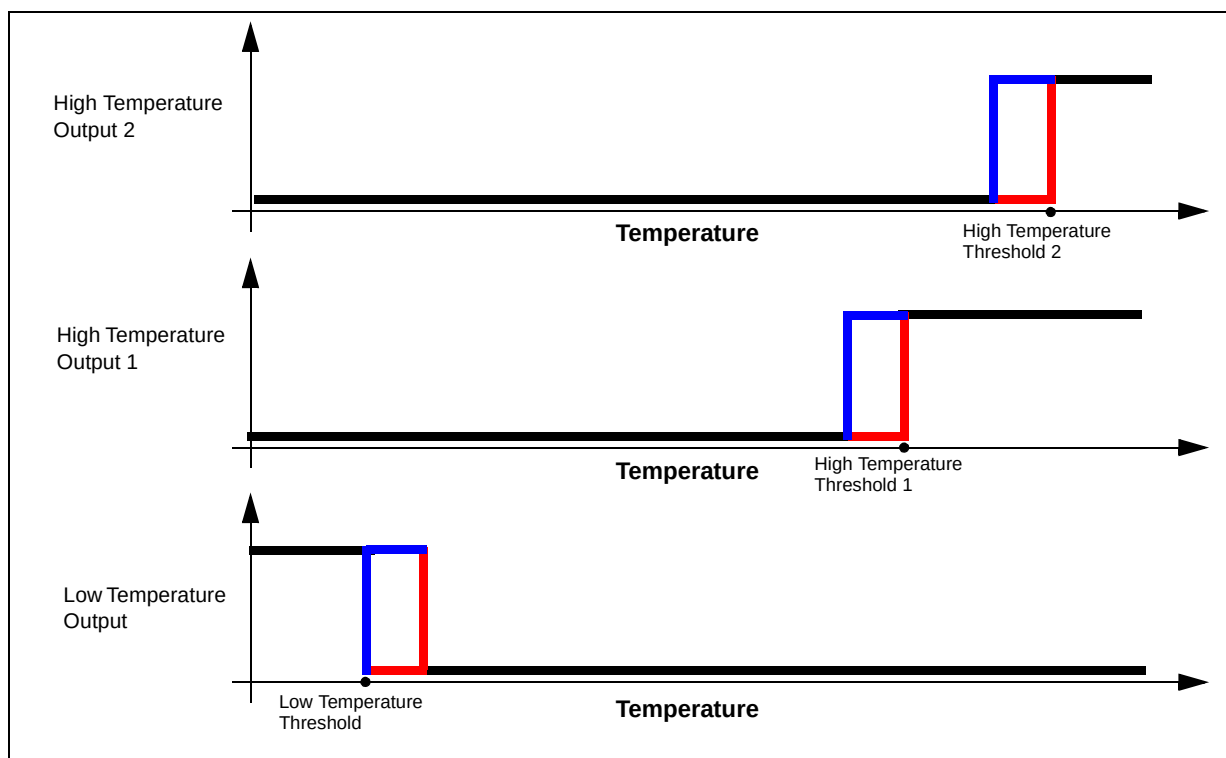


Figure 510. Digital output behavior with temperature

41.2.2 Linear temperature sensor (analog output generation)

When analog output generation is enabled, the temperature sensor outputs a voltage proportional to the internal junction temperature of the chip. This analog voltage signal is converted into a digital code by an onboard ADC. The temperature value is obtained from a

linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programmed into flash memory.

41.3 Temperature formula

The system chain that translates device junction temperature into a digital variable is comprised of the temperature sensor, a bandgap reference and the onboard ADC. Both the temperature sensor analog output voltage and bandgap reference voltage must be converted by the ADC into digital codes. The temperature formula shown in [Figure 511](#) is used to calculate internal device junction temperature. Note that the formula represents a linear relation between temperature sensor analog output voltage and device junction temperature with calibration adjusted coefficients and ADC reference voltage correction (Bandgap reference terms replaces ADC reference terms in the temperature expression).

Figure 511. Temperature formula

$$T = (232 + TSCA \times 2^{-6}) \times \frac{T_{\text{TSENS_CODE}}(T)}{V_{\text{BG_CODE}}(T)} - (273 + TSCB \times 2^{-4}) \text{ [}^{\circ}\text{C]}$$

Where:

$$TSCA = \left(\frac{T_{\text{HIGH}} - T_{\text{LOW}}}{T_{\text{TSENS_CODE}}(T_{\text{HIGH}}) - T_{\text{TSENS_CODE}}(T_{\text{LOW}})} \times V_{\text{BG_CODE}}(T_{\text{FINAL}}) - 232 \right) \times 2^6$$

$$TSCB = \left(\frac{T_{\text{HIGH}} - T_{\text{LOW}}}{T_{\text{TSENS_CODE}}(T_{\text{HIGH}}) - T_{\text{TSENS_CODE}}(T_{\text{LOW}})} \times T_{\text{TSENS_CODE}}(T_{\text{LOW}}) - T_{\text{LOW}} - 273 \right) \times 2^4$$

$$T_{\text{TSENS_CODE}}(T_{\text{LOW}}) = \frac{V_{\text{TSENS}}(T_{\text{LOW}})}{V_{\text{ref0}}} \times 2^{12}$$

$$T_{\text{TSENS_CODE}}(T_{\text{HIGH}}) = \frac{V_{\text{TSENS}}(T_{\text{HIGH}})}{V_{\text{ref0}}} \times 2^{12}$$

$$V_{\text{BG_CODE}}(T_{\text{FINAL}}) = \frac{V_{\text{BG}}(T_{\text{FINAL}})}{V_{\text{ref0}}} \times 2^{12}$$

$$V_{\text{BG_CODE}}(T) = \frac{V_{\text{BG}}(T)}{V_{\text{ref}}} \times 2^{12}$$

$$T_{\text{TSENS_CODE}}(T) = \frac{V_{\text{TSENS}}(T)}{V_{\text{ref}}} \times 2^{12}$$

Notes:

- TSCA and TSCB are 16-bit signed integer constants stored in flash memory during factory test and calibration
- $V_{\text{TSENS}}(T)$ is the temperature sensor output sampled by the ADC
- $V_{\text{BG}}(T)$ is the bandgap voltage sampled by the ADC at temperature T
- V_{ref} is the ADC reference voltage in normal usage of the device
- V_{ref0} is the ADC reference voltage during factory calibration
- T_{LOW} is the low temperature factory calibration temperature
- T_{HIGH} is the hot temperature factory calibration temperature
- T_{FINAL} is the final insertion factory calibration temperature

Parameters TSCA and TSCB are 16-bit signed integer constants stored during factory test and calibration that need to be used in the temperature formula to calculate device junction temperature. The values for TSCA and TSCB translate into positive or negative shifts of the linear relation coefficients that compensate for manufacturing process variation. [Figure 511](#) shows how parameters TSCA and TSCB are calculated in terms of the actual variables measured during factory calibration.

The calibration points are illustrated in [Figure 512](#).

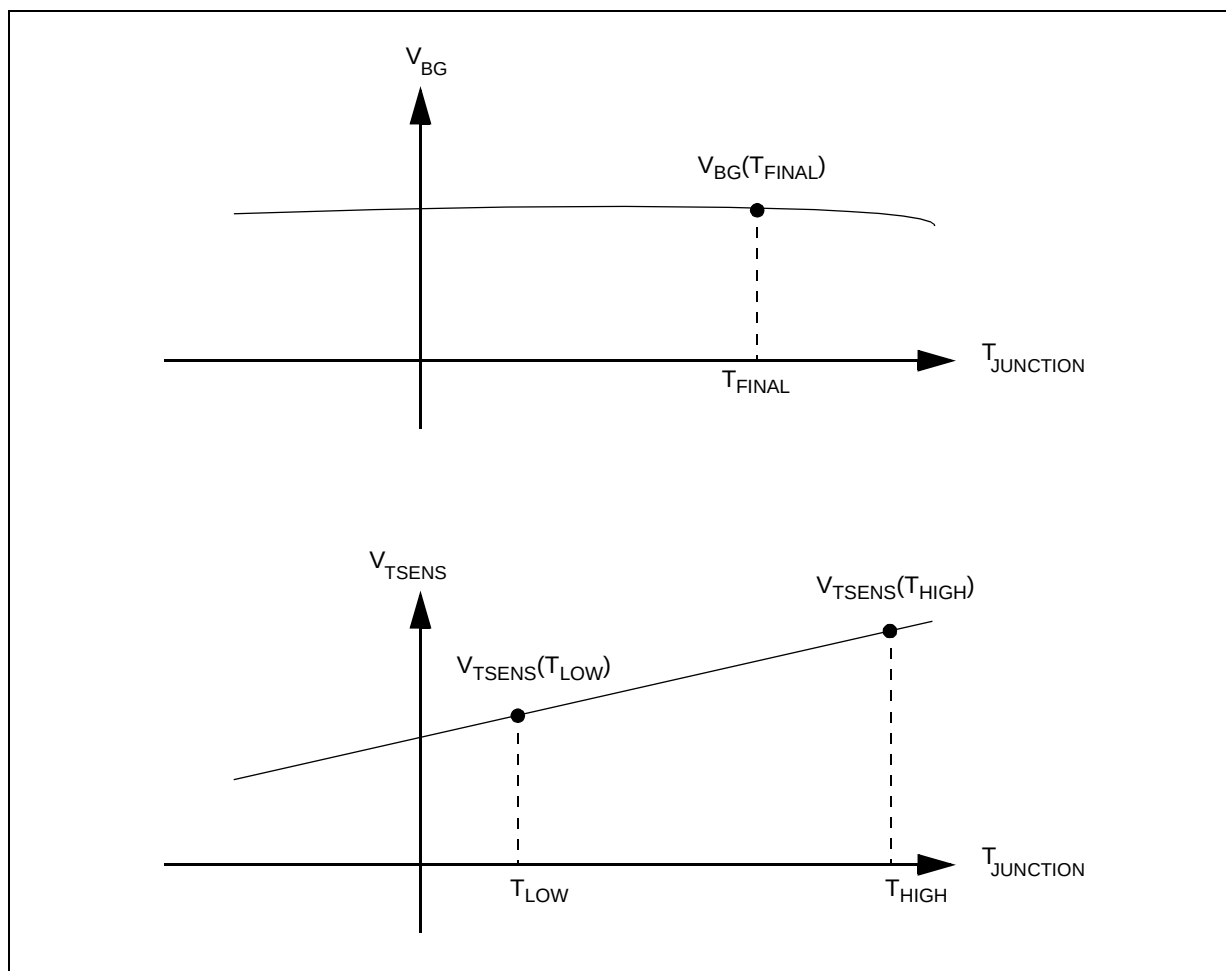


Figure 512. Calibration Points

41.4 Calculating device temperature

Software must perform the following steps in order to calculate device temperature:

1. Measure the bandgap voltage using the onboard ADC module and calculate $V_{BG_CODE}(T)$ according to the formula shown in [Figure 511](#).
2. Measure the temperature sensor output voltage using the onboard ADC module and calculate $T_{TSENS_CODE}(T)$ according to the formula shown in [Figure 511](#).
3. Retrieve parameters TSCA and TSCB from flash memory.
4. Calculate T according to the formula shown in [Figure 511](#).

Note: Software calculation must be made with negligible precision loss associated with limited bit representation of intermediate calculated results.

42 Timers Overview

Three types of timers are implemented on SPC57EM80xx and a complex timer subsystem.

Table 603. Timers

Timer	Device-specific information	Block details
System Timer Module (STM)	Section 7.7.1: System Timer Module (STM) configuration	Chapter 43: System Timer Module (STM)
Software Watchdog Timer (SWT)	Section 7.7.2: Software Watchdog Timer (SWT) configuration	Chapter 44: Software Watchdog Timer (SWT)
Periodic Interrupt Timer (PIT)	Section 7.7.3: PIT configuration	Chapter 45: Periodic Interrupt Timer (PIT)
GTM103 Wrapper Module Configuration ⁽¹⁾	Section 7.7.4: GTM 103 Integration Module (GTMINT103) configuration	Chapter 88: GTM103 Integration (GTMINT) Module

1. The GTM specification is a separate document that is included in the SPC57EM80xx documentation package.

Each core has a dedicated System Timer Module (STM) and a dedicated Software Watchdog Timer (SWT). The checker core does not have a dedicated STM.

The Secure SWT is on the Hardware Security Module (HSM). It is an enhanced version of the SWT that has the ability to trigger an interrupt not only based on some number of clock cycles, but also based on code executing at some user-defined address. For more information, refer to the *SPC57EM80xx Security Manual*.

Two Periodic Interrupt Timer (PIT) module instances are implemented. One PIT module is for 64-bit timer implementation by way of two 32-bit timer channels (0 and 1) with chaining mode; the timer value is unchanged by a non-destructive reset. The other PIT module is for general use and has eight 32-bit timer channels with no chaining mode.

The GTM103 is an implementation of the Robert Bosch GmbH GTM timer subsystem. It is not a single module; it is a complex timer subsystem that consists of many modules. It can be used to implement highly complex timer functions.

43 System Timer Module (STM)

43.1 Introduction

43.1.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

43.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

43.1.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

43.2 External signal description

The STM does not have any external interface signals.

43.3 Memory map and register definition

The STM programming model has fourteen 32-bit registers. The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

43.3.1 Memory map

The STM memory map is shown in [Table 604](#).

Table 604. STM memory map

Address offset	Register	Location
0x0000	STM Control Register (STM_CR)	on page 1343
0x0004	STM Counter Value (STM_CNT)	on page 1344
0x0008	Reserved	
0x000C	Reserved	

Table 604. STM memory map (continued)

Address offset	Register	Location
0x0010	STM Channel 0 Control Register (STM_CCR0)	on page 1345
0x0014	STM Channel 0 Interrupt Register (STM_CIR0)	on page 1345
0x0018	STM Channel 0 Compare Register (STM_CMP0)	on page 1346
0x001C	Reserved	
0x0020	STM Channel 1 Control Register (STM_CCR1)	on page 1345
0x0024	STM Channel 1 Interrupt Register (STM_CIR1)	on page 1345
0x0028	STM Channel 1 Compare Register (STM_CMP1)	on page 1346
0x002C	Reserved	
0x0030	STM Channel 2 Control Register (STM_CCR2)	on page 1345
0x0034	STM Channel 2 Interrupt Register (STM_CIR2)	on page 1345
0x0038	STM Channel 2 Compare Register (STM_CMP2)	on page 1346
0x003C	Reserved	
0x0040	STM Channel 3 Control Register (STM_CCR3)	on page 1345
0x0044	STM Channel 3 Interrupt Register (STM_CIR3)	on page 1345
0x0048	STM Channel 3 Compare Register (STM_CMP3)	on page 1346
0x004C - 0x3FFF	Reserved	

43.3.2 Register descriptions

The following sections detail the individual registers within the STM programming model.

43.3.2.1 STM Control Register (STM_CR)

The STM Control Register (STM_CR) includes the prescale value, freeze control and timer enable bits.

Figure 513. STM Control Register (STM_CR)

Offset: 0x000

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	CPS								0	0	0	0	0	0	FRZ		TEN
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 605. STM_CR field descriptions

Field	Description
16:23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2 ... 0xFF Divide system clock by 256
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
31 TEN	Timer Counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

43.3.2.2 STM Count Register (STM_CNT)

The STM Count Register (STM_CNT) holds the timer count value.

Figure 514. STM Count Register (STM_CNT)

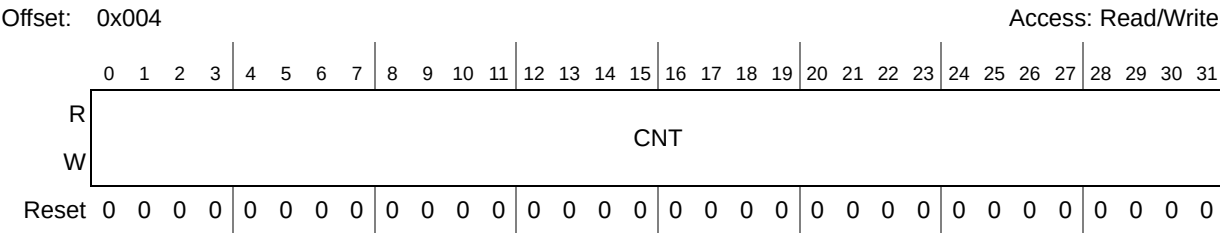


Table 606. STM_CNT field descriptions

Field	Description
0:31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

43.3.2.3 STM Channel Control Register (STM_CCRn)

The STM Channel Control Register (STM_CCRn) has the enable bit for channel n of the timer.

Figure 515. STM Channel Control Register (STM_CCRn)

Offset: 0x10+0x10*n

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 607. STM_CCRn field descriptions

Field	Description
31 CEN	Channel Enable. 0 The channel is disabled. 1 The channel is enabled.

43.3.2.4 STM Channel Interrupt Register (STM_CIRn)

The STM Channel Interrupt Register (STM_CIRn) has the interrupt flag for channel n of the timer.

Figure 516. STM Channel Interrupt Register (STM_CIRn)

Offset: 0x14+0x10*n

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 608. STM_CIRn field descriptions

Field	Description
31 CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

43.3.2.5 STM Channel Compare Register (STM_CMPn)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Figure 517. STM Channel Compare Register (STM_CMPn)

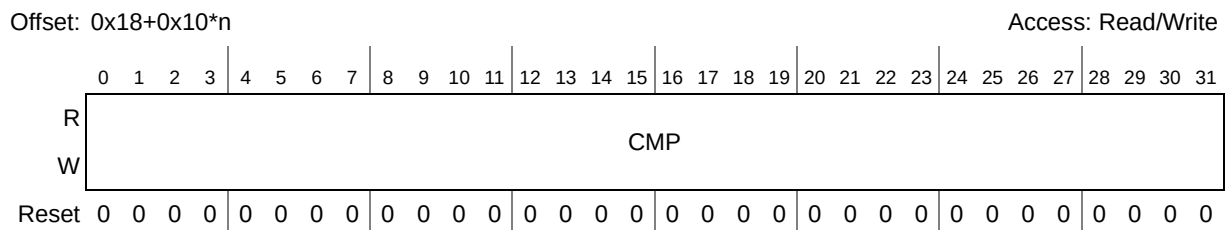


Table 609. STM_CMPn field descriptions

Field	Description
0:31 CMP	Compare value for channel n. If the STM_CCRn[CEN] bit is set and the STM_CMPn register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIRn[CIF] bit is set.

43.4 Functional description

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the system clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode, otherwise it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCRn), a channel interrupt register (STM_CIRn) and a channel compare register (STM_CMPn). The channel is enabled by setting the STM_CCRn[CEN] bit. When enabled, the channel will set the STM_CIRn[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing a 1 to the STM_CIRn[CIF] bit. A write of 0 to the STM_CIRn[CIF] bit has no effect.

Note: The STM counter does not advance when the system clock is stopped.

44 Software Watchdog Timer (SWT)

44.1 Introduction

44.1.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out.

44.1.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of system or oscillator clock, for timer operation
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits

44.1.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR [on page 1348](#). If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit in the SWT_CR [on page 1348](#). If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

44.2 External signal description

The SWT module does not have any external interface signals.

44.3 Memory map and register definition

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access, otherwise a bus error is generated. If either the HLK or

SLK bits in the SWT_CR are set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

The SWT memory map is shown in [Table 610](#). The reset values of SWT_CR and SWT_TO are device specific. These values are determined by SWT inputs.

Table 610. SWT memory map

Offset	Register	Access	Reset Value	Location
0x0000	SWT Control Register (SWT_CR)	R/W	— ⁽¹⁾	on page 1348
0x0004	SWT Interrupt Register (SWT_IR)	R/W	0x0000_0000	on page 1350
0x0008	SWT Time-out Register (SWT_TO)	R/W	— ⁽¹⁾	on page 1350
0x000C	SWT Window Register (SWT_WN)	R/W	0x0000_0000	on page 1351
0x0010	SWT Service Register (SWT_SR)	W	0x0000_0000	on page 1351
0x0014	SWT Counter Output Register (SWT_CO)	R	0x0000_0000	on page 1352
0x0018	SWT Service Key Register (SWT_SK)	R/W	0x0000_0000	on page 1352
0x001C-0x3FFF	Reserved			

1. The reset value for this register is device specific. See the device configuration chapter.

44.3.1 Register descriptions

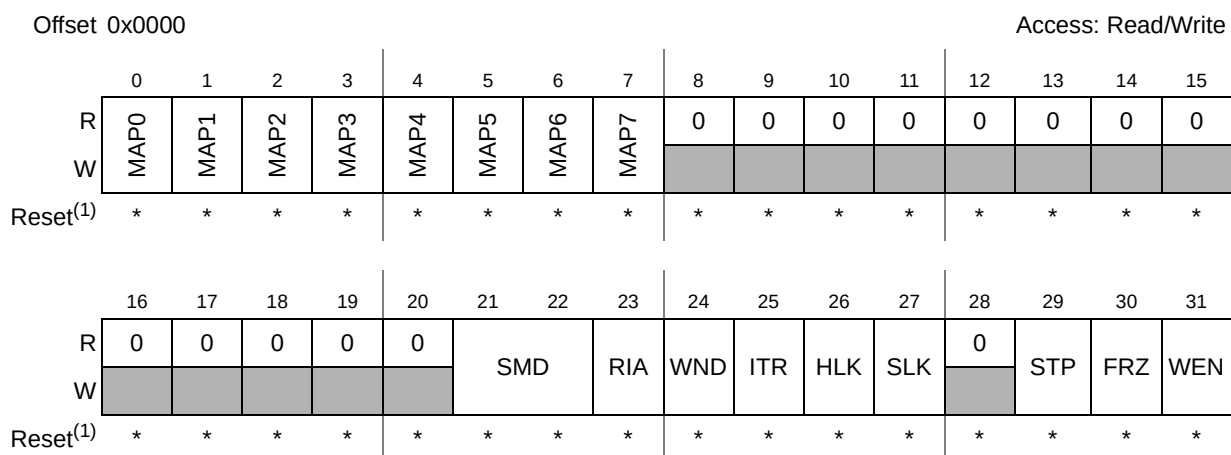
The following sections detail the individual registers within the SWT programming model.

Note: See the memory map for the base address of SWT registers.

44.3.1.1 SWT Control Register (SWT_CR)

The SWT_CR contains fields for configuring and controlling the SWT. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Figure 518. SWT Control Register (SWT_CR)



1. The reset value for the SWT_CR is device specific. See the device configuration section.

Table 611. SWT_CR field descriptions

Field	Name	Description
0-7	MAPn	Master Access Protection for Master n. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
21-22	SMD	Service Mode. 00 Fixed Service Sequence, the watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to the SWT_SR. 01 Keyed Service Sequence, the watchdog is serviced by writing two pseudorandom key values to the SWT_SR. 10 Fixed Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register which can not be updated while the watchdog is enabled. Support for this mode is device and instance specific. 11 Incremental Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register which can be updated. Support for this mode is device and instance specific.
23	RIA	Reset on Invalid Access. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a system reset if WEN=1
24	WND	Window Mode. 0 Regular mode, service sequence can be done at any time 1 Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
25	ITR	Interrupt Then Reset. 0 Generate a reset on a time-out 1 Generate an interrupt on an initial time-out, reset on a second consecutive time-out
26	HLK	Hard Lock. This bit is only cleared at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27	SLK	Soft Lock. This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
29	STP	Stop Mode Control. Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30	FRZ	Debug Mode Control. Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31	WEN	Watchdog Enabled. 0 SWT is disabled 1 SWT is enabled

44.3.1.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Figure 519. SWT Interrupt Register (SWT_IR)

Offset	0x0004																Access: Read/Write
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF	
W																w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 612. SWT_IR field descriptions

Field	Name	Description
31	TIF	Time-out Interrupt Flag. The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request. 1 Interrupt request due to an initial time-out.

44.3.1.3 SWT Time-Out Register (SWT_TO)

The SWT Time-Out (SWT_TO) register contains the 32-bit time-out period. The reset value for this register is device specific. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Figure 520. SWT Time-Out Register (SWT_TO)

Offset	0x008																												Access: Read/Write			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WTO																															
W																																
Reset ⁽¹⁾	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

1. The reset value of the SWT_TO register is device specific. See the device configuration section.

Table 613. SWT_TO field descriptions

Field	Name	Description
0-31	WTO	Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with this value when the service sequence is written or when the SWT is enabled. The initial value is 0x0003_FDE0. The minimum value is 0x0000_0100.

44.3.1.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Figure 521. SWT Window Register (SWT_WN)

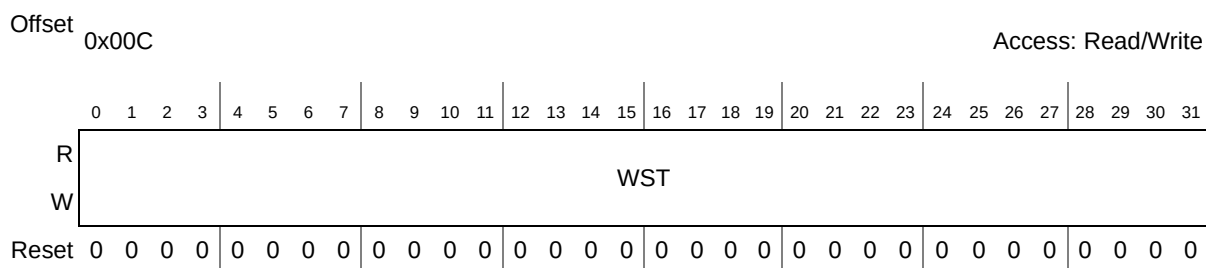


Table 614. SWT_WN field descriptions

Field	Name	Description
0-31	WST	Window start value. When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

44.3.1.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Figure 522. SWT Service Register (SWT_SR)

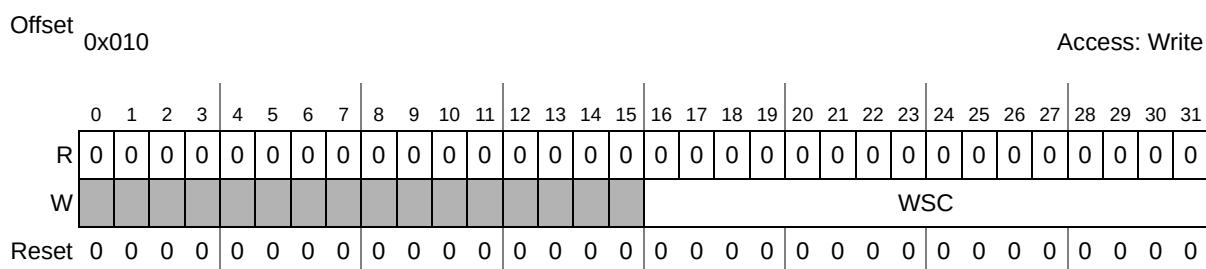


Table 615. SWT_SR field descriptions

Field	Name	Description
16-31	WSC	Watchdog Service Code. This field is used to service the watchdog and to clear the soft lock bit (SWT_CR[SLK]). If the SWT_CR[KEY] bit is set, two pseudorandom key values are written to service the watchdog, see Section (44.3.1.1) for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

44.3.1.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Figure 523. SWT Counter Output Register (SWT_CO)

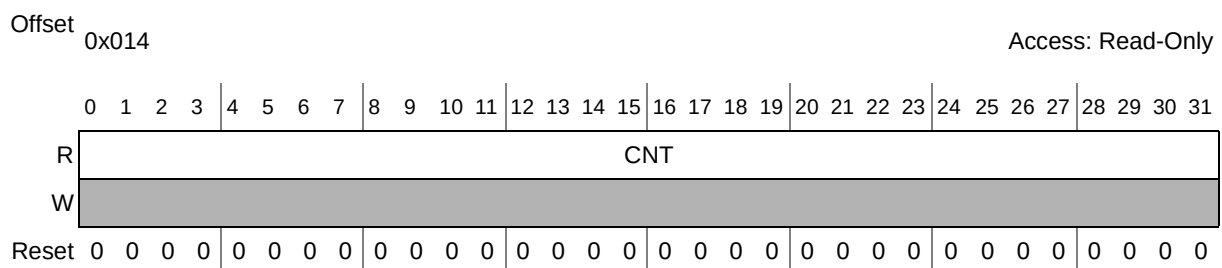


Table 616. SWT_CO field descriptions

Field	Name	Description
0-31	CNT	Watchdog Count. When the watchdog is disabled (SWT_CR[WEN]=0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN]=1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

44.3.1.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HCLK] or SWT_CR[SLK] bits are set.

Figure 524. SWT Service Key Register (SWT_SK)

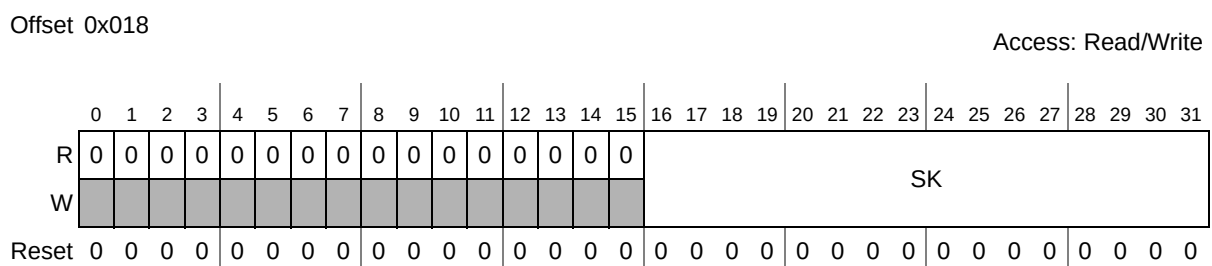


Table 617. SWT_SK field descriptions

Field	Name	Description
16-31	SK	Service Key. This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[KEY] is set, the next key value to be written to the SWT_SR is $(17 \times SK + 3) \bmod 2^{16}$.

44.4 Functional description

The SWT is a 32-bit timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes a control register (SWT_CR), an interrupt register (SWT_IR), a time-out register (SWT_TO), a window register (SWT_WN), a service register (SWT_SR), a counter output register (SWT_CO) and a service key register (SWT_SK).

The SWT_CR register includes bits to enable the timer, set configuration options and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific^(m). If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100 in which case the time-out period is set to 0x100. When the SWT is enabled, this time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed. The SWT_CR[CSL] bit selects which clock (system or oscillator) is used to drive the down counter. The reset value of the SWT_TO register is device specific.

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only. The hard lock is enabled by setting the SWT_CR[HCLK] bit, which can only be cleared by a reset. The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register. The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

When enabled, the SWT requires periodic execution of a servicing operation which is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- fixed service sequence
- keyed service sequence
- fixed execution address
- incremental execution address

(Some devices or instances only support the fixed service sequence and keyed service sequence modes.) If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected which requires writing 0xA602, 0xB480 to the SWT_SR[WSC] field to service the

m. See SWT section in device configuration chapter.

watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes. If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected which requires writing two pseudorandom keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudorandom key generator defined in [Figure 525](#). This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

$$SK_{n+1} = (17 * SK_n + 3) \bmod 2^{16}$$

Figure 525. Pseudorandom Key Generator

If the SWT_CR[SMD] field is set to 10b, the fixed execution address mode is selected which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can not be updated while the watchdog is enabled. If the SWT_CR[SMD] field is set to 11b, the incremental execution address mode is selected in which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can be updated while the watchdog is enabled.

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

If window mode is enabled (SWT_CR[WND] bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_CR[RIA] bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to 1000 then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

The interrupt then reset bit (SWT_CR[ITR]) controls the action taken when a time-out occurs. If the SWT_CR[ITR] bit is not set, a reset is generated immediately on a time-out. If the SWT_CR[ITR] bit is set, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period. If the service sequence is not written before the second consecutive time-out, the SWT generates a system reset. The interrupt is indicated by the time-out interrupt flag (SWT_IR[TIF]). The interrupt request is cleared by writing a one to the SWT_IR[TIF] bit.

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

45 Periodic Interrupt Timer (PIT)

45.1 Introduction

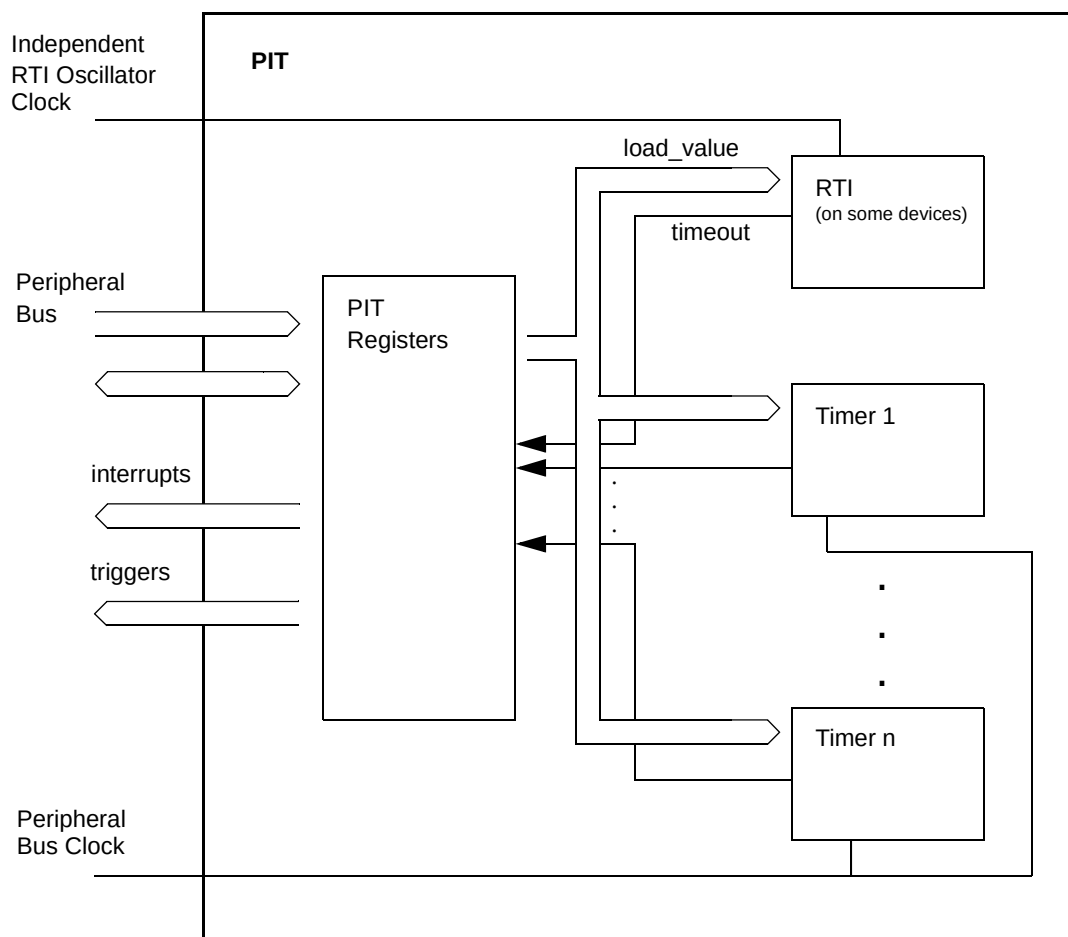
The PIT timer module is an array of timers that can be used to raise interrupts and trigger DMA channels. It includes a dedicated Real Time Interrupt Timer (RTI), which runs on a separate clock and can be used for system wakeup.

Note: For the chip-specific implementation details of this module's instances see the *Device Configuration* chapter.

45.1.1 Block diagram

[Table 526](#) shows the PIT block diagram.

Figure 526. Block diagram of the PIT



Note: Refer to the *Chip Configuration* information for the number of PIT channels used in this MCU.

45.1.2 Features

The main features of this block are:

- One RTI (Real-Time Interrupt) timer to wake up the CPU in Stop mode
- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable
- RTI interrupt can be raised, even when the bus clock is switched off
- Power saving with a separate input clock for the RTI timer (all other timers share one common core clock)
- Independent timeout periods for each timer

45.2 Signal description

This module has no external pins.

45.2.1 Memory map/register description

This section provides a detailed description of all registers accessible in this module.

Note: Reserved registers will read as 0, writes will have no effect.

Note: Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

Note: The RTI registers should be programmed only when the RTI clock is running.

Table 618. PITx memory map

Offset (hex)	Register name	Width (bits)	Access	Location
0000_0000	PIT Control Register (PITx_MCR)	32	R/W	on page 1359
0000_0004–0000_00DF	Reserved			
0000_00E0	PIT Upper Lifetime Timer Register (PITx_LTMR64H)	32	R	on page 1360
0000_00E4	PIT Lower Lifetime Timer Register (PITx_LTMR64L)	32	R	on page 1360
0000_00F0	PIT RTI Timer Load Value Register (PITx_RTI_LDVAL)	32	R/W	on page 1364
0000_00F4	PIT RTI Current Timer Value Register (PITx_RTI_CVAL)	32	R	on page 1365
0000_00F8	PIT RTI Timer Control Register (PITx_RTI_TCTRL)	32	R/W	on page 1366
0000_00FC	PIT RTI Timer Flag Register (PITx_RTI_TFLG)	32	R/W	on page 1367
0000_0100	PIT Timer Load Value Register 0 (PITx_LDVAL0)	32	R/W	on page 1364
0000_0104	PIT Current Timer Value Register 0 (PITx_CVAL0)	32	R	on page 1365
0000_0108	PIT Timer Control Register 0 (PITx_TCTRL0)	32	R/W	on page 1366
0000_010C	PIT Timer Flag Register 0 (PITx_TFLG0)	32	R/W	on page 1367
0000_0110	PIT Timer Load Value Register 1 (PITx_LDVAL1)	32	R/W	on page 1364

Table 618. PITx memory map (continued)

Offset (hex)	Register name	Width (bits)	Access	Location
0000_0114	PIT Current Timer Value Register 1 (PITx_CVAL1)	32	R	on page 1365
0000_0118	PIT Timer Control Register 1 (PITx_TCTRL1)	32	R/W	on page 1366
0000_011C	PIT Timer Flag Register 1 (PITx_TFLG1)	32	R/W	on page 1367
0000_0120	PIT Timer Load Value Register 2 (PITx_LDVAL2)	32	R/W	on page 1364
0000_0124	PIT Current Timer Value Register 2 (PITx_CVAL2)	32	R	on page 1365
0000_0128	PIT Timer Control Register 2 (PITx_TCTRL2)	32	R/W	on page 1366
0000_012C	PIT Timer Flag Register 2 (PITx_TFLG2)	32	R/W	on page 1367
0000_0130	PIT Timer Load Value Register 3 (PITx_LDVAL3)	32	R/W	on page 1364
0000_0134	PIT Current Timer Value Register 3 (PITx_CVAL3)	32	R	on page 1365
0000_0138	PIT Timer Control Register 3 (PITx_TCTRL3)	32	R/W	on page 1366
0000_013C	PIT Timer Flag Register 3 (PITx_TFLG3)	32	R/W	on page 1367
0000_0140	PIT Timer Load Value Register 4 (PITx_LDVAL4)	32	R/W	on page 1364
0000_0144	PIT Current Timer Value Register 4 (PITx_CVAL4)	32	R	on page 1365
0000_0148	PIT Timer Control Register 4 (PITx_TCTRL4)	32	R/W	on page 1366
0000_014C	PIT Timer Flag Register 4 (PITx_TFLG4)	32	R/W	on page 1367
0000_0150	PIT Timer Load Value Register 5 (PITx_LDVAL5)	32	R/W	on page 1364
0000_0154	PIT Current Timer Value Register 5 (PITx_CVAL5)	32	R	on page 1365
0000_0158	PIT Timer Control Register 5 (PITx_TCTRL5)	32	R/W	on page 1366
0000_015C	PIT Timer Flag Register 5 (PITx_TFLG5)	32	R/W	on page 1367
0000_0160	PIT Timer Load Value Register 6 (PITx_LDVAL6)	32	R/W	on page 1364
0000_0164	PIT Current Timer Value Register 6 (PITx_CVAL6)	32	R	on page 1365
0000_0168	PIT Timer Control Register 6 (PITx_TCTRL6)	32	R/W	on page 1366
0000_016C	PIT Timer Flag Register 6 (PITx_TFLG6)	32	R/W	on page 1367
0000_0170	PIT Timer Load Value Register 7 (PITx_LDVAL7)	32	R/W	on page 1364
0000_0174	PIT Current Timer Value Register 7 (PITx_CVAL7)	32	R	on page 1365
0000_0178	PIT Timer Control Register 7 (PITx_TCTRL7)	32	R/W	on page 1366
0000_017C	PIT Timer Flag Register 7 (PITx_TFLG7)	32	R/W	on page 1367

45.2.1.1 PIT Module x Control Register (PITx_MCR)

The PIT Module Control Register (PITx_MCR) controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Figure 527. PIT Module Control Register (PITx_MCR)

Address: Base + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MDIS_RTI	MDIS	FRZ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Table 619. PITx_MCR field descriptions

Field	Description
0–28 Reserved	This read-only bitfield is reserved and always has the value zero.
29 MDIS_RTI	Module Disable - RTI section This is used to disable the RTI timer. This bit must be enabled before any RTI setup is done. 0 Clock for RTI timers is enabled. 1 Clock for RTI timers is disabled.
30 MDIS	Module Disable This is used to disable the module clock. The RTI timer is not affected by this bit. This bit must be enabled before any other setup is done. 0 Clock for PIT timers is enabled. 1 Clock for PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters debug mode. 0 Timers continue to run in debug mode. 1 Timers are stopped in debug mode.

45.2.1.2 PIT Module x Upper Lifetime Timer Register (PITx_LTMR64H)

The PIT Module x Upper Lifetime Timer Register (PITx_LTMR64H) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

Figure 528. PIT Upper Lifetime Timer Register (PITx_LTMR64H)

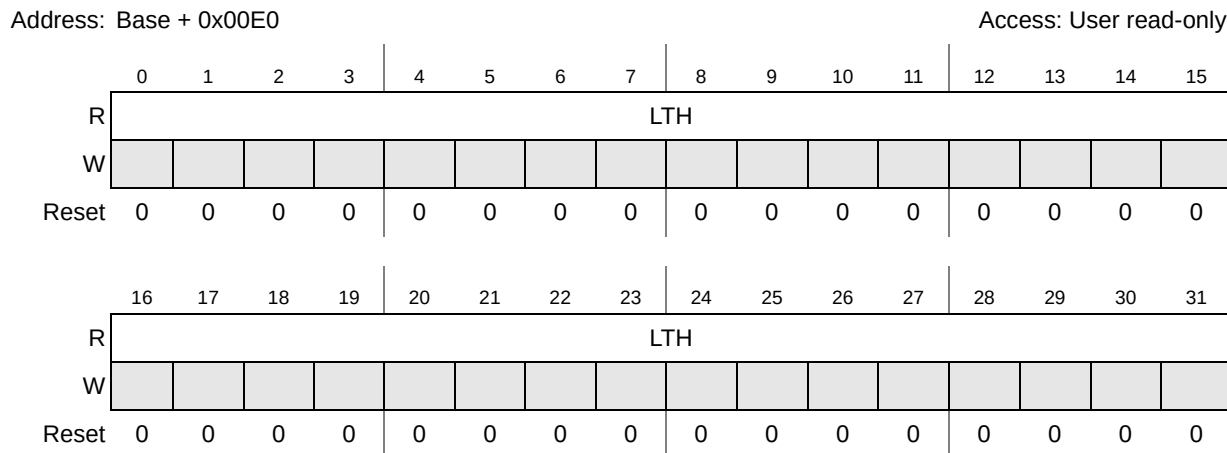


Table 620. PITx_LTMR64H field descriptions

Field	Description
0–31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

45.2.1.3 PIT Module x Lower Lifetime Timer Register (PITx_LTMR64L)

The PIT Module x Lower Lifetime Timer Register (PITx_LTMR64L) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Figure 529. PIT Lower Lifetime Timer Register (PITx_LTMR64L)

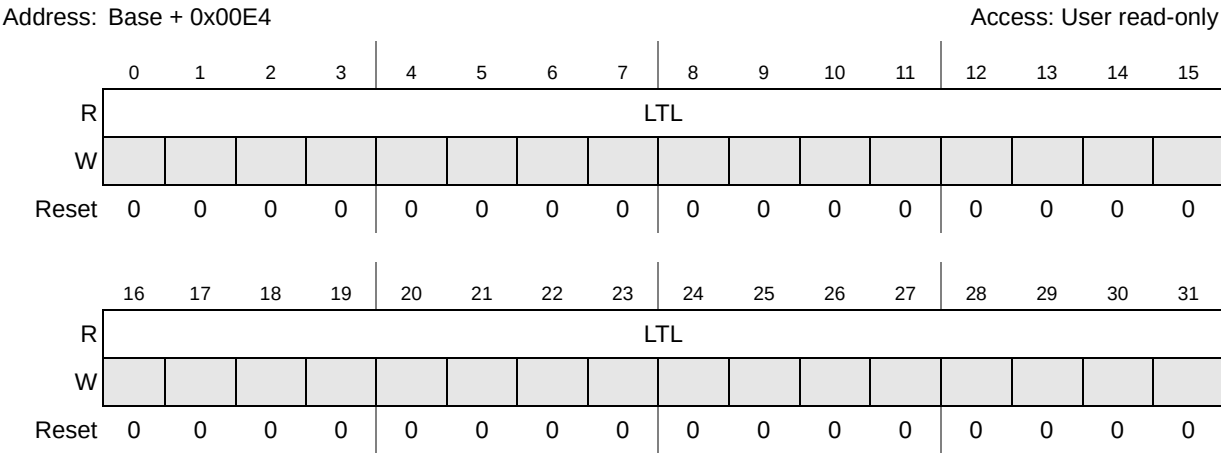


Table 621. PITx_LTMR64L field descriptions

Field	Description
0–31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

45.2.1.4 PIT Module x RTI Timer Load Value Register (PITx_RTI_LDVAL)

This register select the timeout period for the timer interrupts.

In the case of the RTI, it takes several cycles until this value is synchronized into the RTI clock domain. For all other timers the value change is visible immediately. The synchronization mechanism allows 0 wait states in this case.

Figure 530. PIT Module x Timer Load Value Register n (PITx_LDVALn)

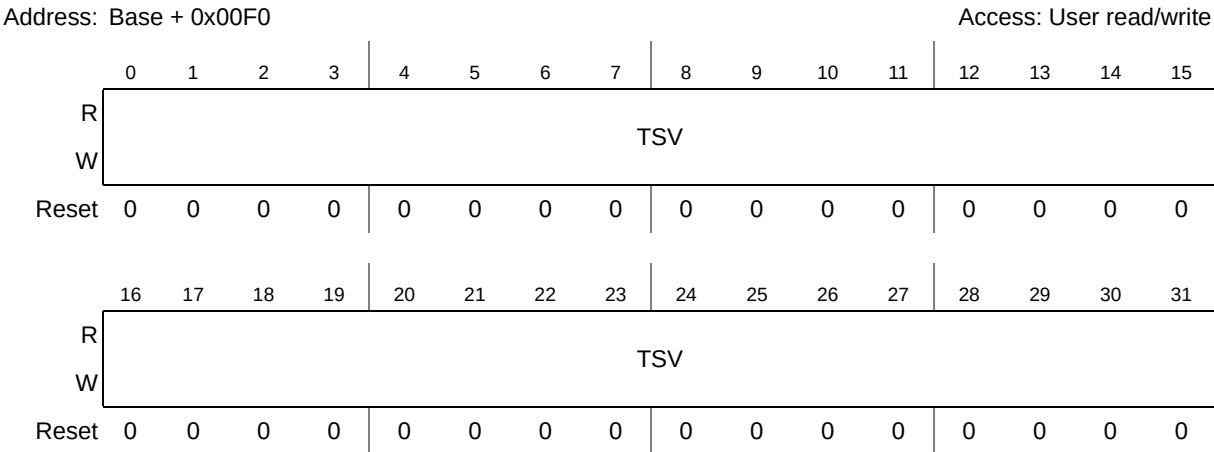


Table 622. PITx_LDVALn field descriptions

Field	Description
0–31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. <i>Note: The RTI timer must not be set to a value lower than 32 cycles, otherwise interrupts may be lost, as it takes several cycles to clear the RTI interrupt. For the other timers, this limit does not apply, however there will be practical limits, since the processor will require several cycles to service an interrupt.</i>

45.2.1.5 PIT Module x RTI Current Timer Value Register (PITx_RTI_CVAL)

This register indicate the current timer position.

For the RTI timers, it shows a value which is several cycles old, since it originates from a potentially different clock domain.

Figure 531. PIT Module x Current Timer Value Register n (PITx_CVALn)

Address: Base + 0x00F4

Access: User read-only

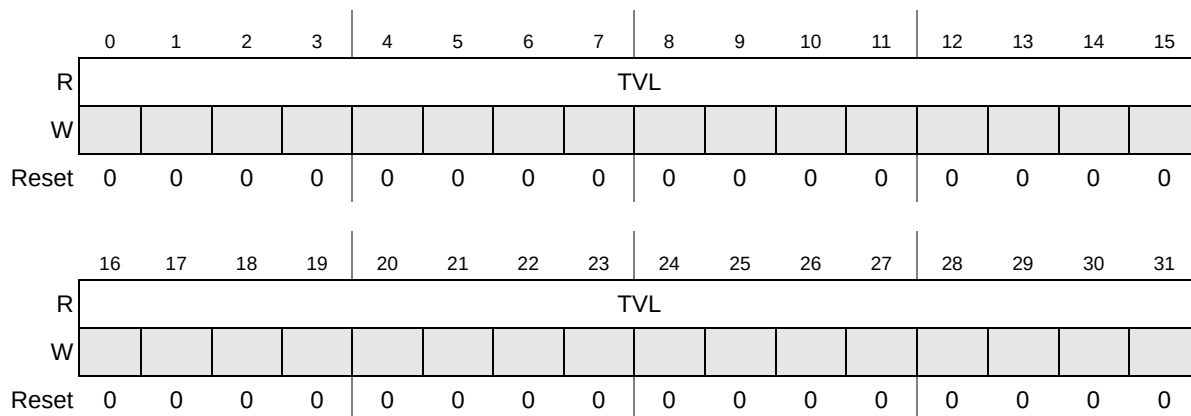


Table 623. PITx_CVALn field descriptions

Field	Description
0–31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. <i>Note: The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.</i>

45.2.1.6 PIT Module x RTI Timer Control Register (PITx_RTI_TCTRL)

This register contain the control bits for each timer.

Figure 532. Timer Control Register (PITx_TCTRLn)

Address: Base + 0x00F8

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 624. PITx_TCTRLn field descriptions

Field	Description
0–28 Reserved	This read-only bitfield is reserved and always has the value zero.
29 CHN	Chain Mode Bit When activated, timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be changed. 0 Timer is not chained. 1 Timer is chained to previous timer (example, for channel 2 if this bit is set timer 2 is chained 1).
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

45.2.1.7 PIT Module x RTI Timer Flag Register (PITx_RTI_TFLG)

This register holds the PIT interrupt flags.

Figure 533. Timer Flag Register *n* (PITx_TFLGn)

Address: Base + 0x00FC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 625. PITx_TFLGn field descriptions

Field	Description
0–30 Reserved	This read-only bitfield is reserved and always has the value zero.
31 TIF	Timer Interrupt Flag TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

45.2.1.8 PIT Module x Timer Load Value Register *n* (PITx_LDVALn)**Figure 534. PIT Module x Timer Load Value Register *n* (PITx_LDVALn)**

Address: Base + 0x0100

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 626. PITx_LDVALn field descriptions

Field	Description
0–31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

45.2.1.9 PIT Module x Current Timer Value Register n (PITx_CVALn)

Figure 535. PIT Module x Current Timer Value Register n (PITx_CVALn)

Address: Base + 0x0104

Access: User read-only

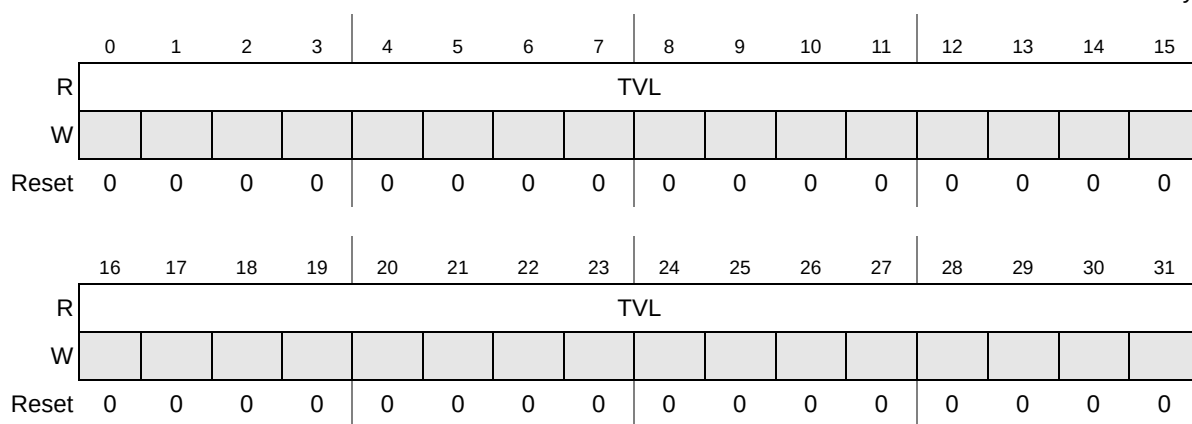


Table 627. PITx_CVALn field descriptions

Field	Description
0–31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. <i>Note: The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.</i>

45.2.1.10 PIT Module x Timer Control Register n (PITx_TCTRLn)

Figure 536. Timer Control Register (PITx_TCTRLn)

Address: Base + 0x0108

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 628. PITx_TCTRLn field descriptions

Field	Description
0–28 Reserved	This read-only bitfield is reserved and always has the value zero.
29 CHN	Chain Mode Bit When activated, timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be changed. 0 Timer is not chained. 1 Timer is chained to previous timer (example, for channel 2 if this bit is set timer2 is chained 1).
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

45.2.1.11 PIT Module x Timer Flag Register n (PITx_TFLGn)

Figure 537. Timer Flag Register n (PITx_TFLGn)

Address: Base + 0x010C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 629. PITx_TFLGn field descriptions

Field	Description
0–30 Reserved	This read-only bitfield is reserved and always has the value zero.
31 TIF	Timer Interrupt Flag TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

45.3 Functional description

This section provides the functional description of the module.

45.3.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts. Each interrupt is available on a separate interrupt line. Additionally the RTI timer can be used to wake up the processor.

45.3.1.1 Timers/RTI

The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared. Since in the case of the RTI, clearing the interrupt crosses clock domains, a minimum load value of 32 should be maintained.

If desired, the current counter value of the timer can be read via the CVAL registers. The value of the RTI counter can be delayed considerably, as it is synchronized to the bus clock from the RTI clock domain.

The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (see the following figure).

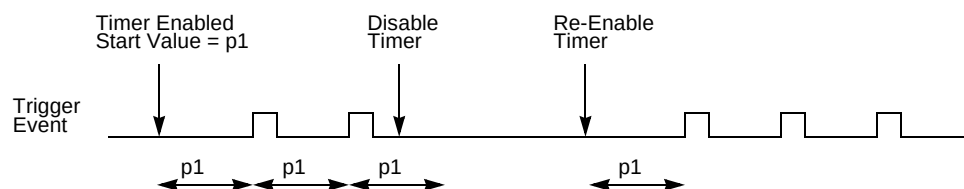


Figure 538. Stopping and starting a timer

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (see the following figure).

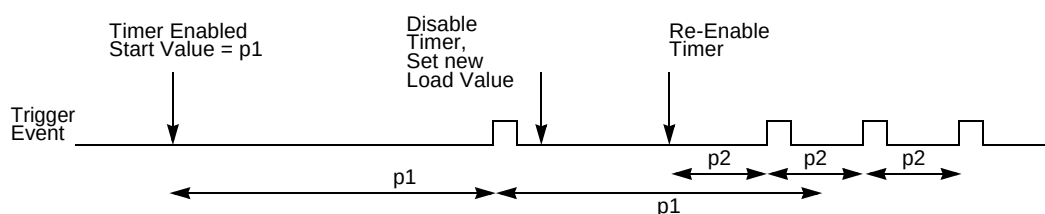


Figure 539. Modifying running timer period

It is also possible to change the counter period without restarting the timer by writing the LDVAL register with the new load value. This value will then be loaded after the next trigger event (see the following figure).

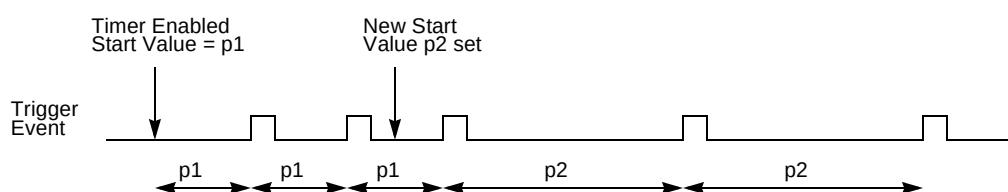


Figure 540. Dynamically setting a new load value

45.3.1.2 Debug mode

In debug mode, the timers will be frozen based on field PITx_MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (for example, the timer values) and then continue the operation.

45.3.2 Interrupts

All of the timers support interrupt generation. The RTI is typically used for system wakeup, but can be used for interrupt generation as well. Refer to the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting the TIE bits. The timer interrupt flags (TIF) are set to 1 when a timeout occurs on the associated timer, and are cleared to '0' by writing a '1' to that TIF bit.

The PIT RTI generates a real time interrupt when the selected interrupt time period elapses. The RTI interrupt is disabled locally by setting the TIE bit to zero. The real time interrupt flag (TIF) is set to '1' when a timeout occurs, and is cleared by writing a '1' to the TIF bit. (The flag will be set regardless whether the interrupt is enabled.)

The RTI can be used for periodic wakeup from a low power mode. It can also be used to generate a general purpose interrupt.

45.3.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer $n-1$ has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

45.4 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- The RTI clock has a frequency of 10 MHz.
- The RTI creates a wakeup interrupt every 500 ms.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

First the PIT module must be activated by writing a 0 to field PITx_MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns and the 10 MHz frequency equates to a clock period of 100 ns. Therefore, the RTI timer needs to trigger every 500 ms/100 ns = 5000000 cycles. Timer 1 needs to trigger every 5.12 ms/20 ns = 256000 cycles and timer 3 every 30 ms/20 ns = 1500000 cycles. The value for the LDVAL register trigger is calculated as:

$LDVAL\ trigger = (period / clock\ period) - 1$

This means RTI_LDVAL will be written with 004C4B3F hex, LDVAL1 should be written with 0x0003E7FF, and LDVAL3 should be written with 0x0016E35F.

To generate the wakeup interrupt, the interrupt line must be enabled by writing a '1' to bit RTI_TCTRL[TIE]. To start the RTI, bit RTI_TCTRL[TEN] must also be set.

The interrupt for Timer 1 is enabled by setting bit TCTRL1[TIE]. The timer is started by writing '1' to bit TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore Timer 3 is started by writing a '1' to bit TCTRL3[TEN]; bit TIE stays at 0.

The following example code matches the described setup:

Example 14

```
// turn on PIT
PIT_MCR = 0x00;

// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE; // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN; // start RTI

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

45.5 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 hour.

First the PIT module needs to be activated by writing a '0' to field PITx_MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to timer 1 and programmed to trigger every 10 times.

The value for the LDVAL register trigger is calculated as number of cycles –1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for timer 2 is enabled by setting PITx_TCTRL2[TIE], the chain mode is activated by setting CHN, and the timer is started by writing a '1' to PITx_TCTRL2[TEN]. In PITx_TCTRL1, the TEN bit needs to be set, and CHN and TIE bits are cleared.

The following example code matches the described setup:

Example 15

```
// turn on PIT
PIT_CTRL = 0x00;

// Timer 2
PIT_LDVAL2 = 0x00000009; // setup timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain timer 2 to timer 1
PIT_TCTRL2 |= TEN; // start timer 2
```

```
// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup timer 1 for 600 000 000 cycles
PIPIT_TCTRL1 = TEN; // start timer 1
```

45.6 Example Configuration for the Lifetime Timer

To configure the life timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a '0' to field PITx_MCR[MDIS], then the LDVAL registers need to be set to the maximum value.

The following example code matches the described setup:

Example 16

```
// turn on PIT
PIT_CTRL = 0x00;

// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 2 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1

// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime read first LTMR64H then LTMR64L.

Equation 26 $\text{current_uptime} = \text{PIT_LTMR64H} \ll 64;$

$\text{current_uptime} = \text{current uptime} + \text{PIT_LTMR64L}$

The timer is a down counter.

46 CAN Subsystem

46.1 Introduction

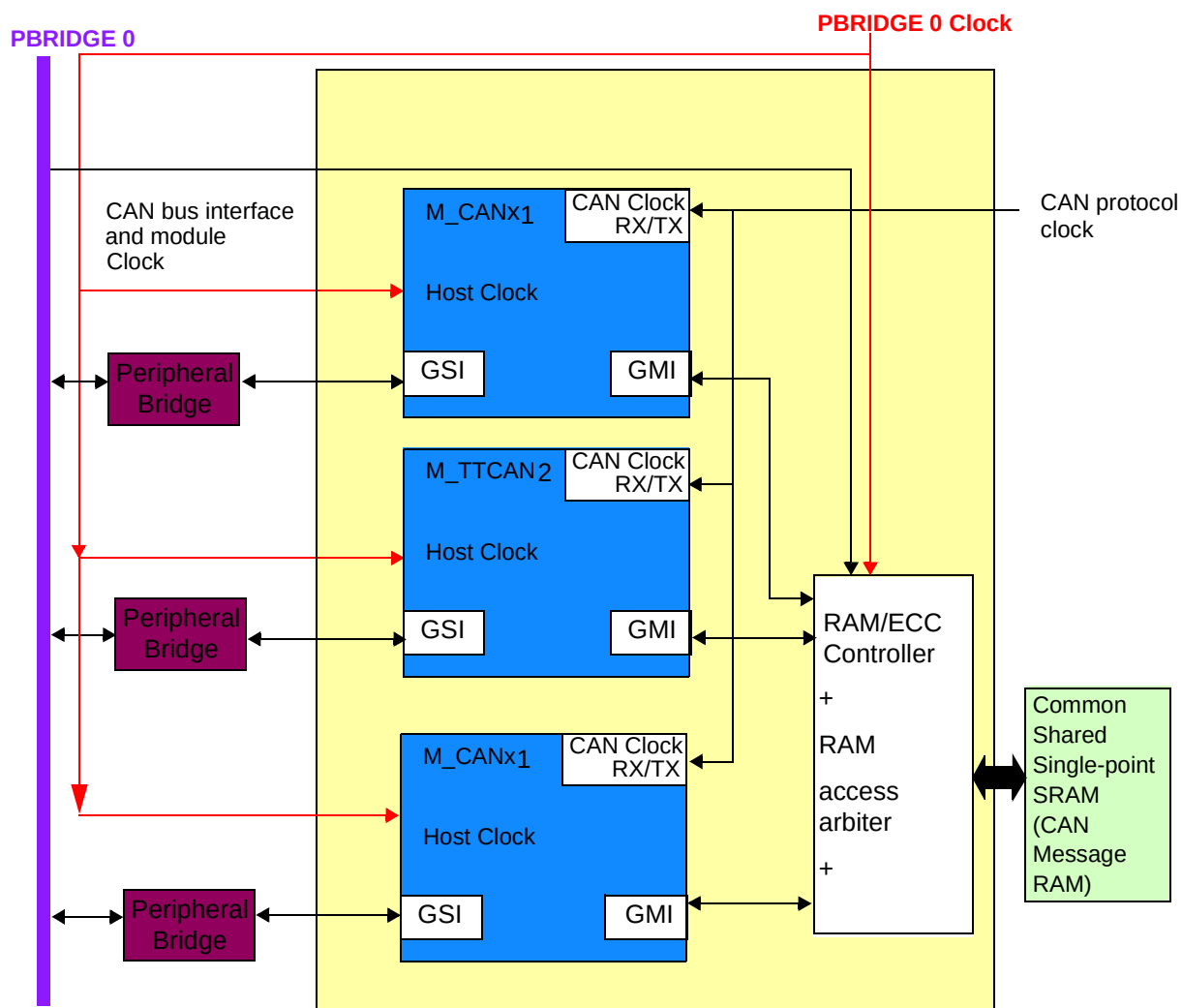
The Controller Area Network (CAN) subsystem consists of the modular CAN (M_CAN) modules and Time triggered CAN (M_TTCAN) modules along with an integrated intelligent CAN RAM controller. The CAN RAM controller consists of additional logic for arbitration between the requests for the RAM access by the various CANs and CPU, ECC encoder/decoder for the Message RAM data and active transmit message buffer protection and M_TTCAN trigger memory from CPU write access. The subsystem follows the little endian format.

Note: Please refer to the Device Configuration chapter to see the number of CAN nodes used in the device.

Table 630. References

IP Name	Version
M_CAN IP	Revision 2.0
M_TTCAN IP	Revision 2.0

The general CAN subsystem implementation is shown in [Figure 541](#). It interfaces with the host processor bus using the peripheral bus. The RAM is not implemented inside the subsystem.



1: Number of M_CAN nodes vary per device. x = 1 to n. Please refer Device Configuration chapter to see the number of M_CAN nodes used in the device.

2: Please refer Device Configuration chapter to see the number of M_TTCAN nodes used in the device.

Figure 541. CAN subsystem generic block diagram

46.2 Features

The CAN subsystem consists of the following major blocks:

- Modular CAN cores: The registers of the CAN module can be accessed using the Generic Slave Interface (GSI)
- Time triggered CAN core. The registers of the CAN module can be accessed using the Generic Slave Interface (GSI)
- CAN-RAM arbiter
- SRAM interface and memory organization
- ECC Controller
- Transmit message buffer locking mechanism
 - Trigger memory locking

46.3 Modular CAN cores

CAN functionality conforms to CAN specification V2.0B active for each CAN node. The M_CAN performs communication according to the CAN protocol specification 2.0 part A,B and to CAN FD 1.0. Flexible assignment of Message Objects to nodes.

The bit rate can be programmed to values up to 1 Mbit/s. Additional transceiver hardware is required for connection to the physical layer.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface. Depending on the chosen device, multiple M_CAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_CAN's clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

46.3.1 Features

The following are the features of Modular CAN cores.

- Conforms with CAN protocol version 2.0 part A, B and ISO 11898-1
- CAN Flexible data-rate (CAN-FD) protocol with maximum 8 data bytes is supported
- Bit rates up to 1 Mbit/s (in standard CAN mode)
- CAN error logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_CANs share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8-/16-/32-bits Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug over M_CAN

46.3.2 Block diagram

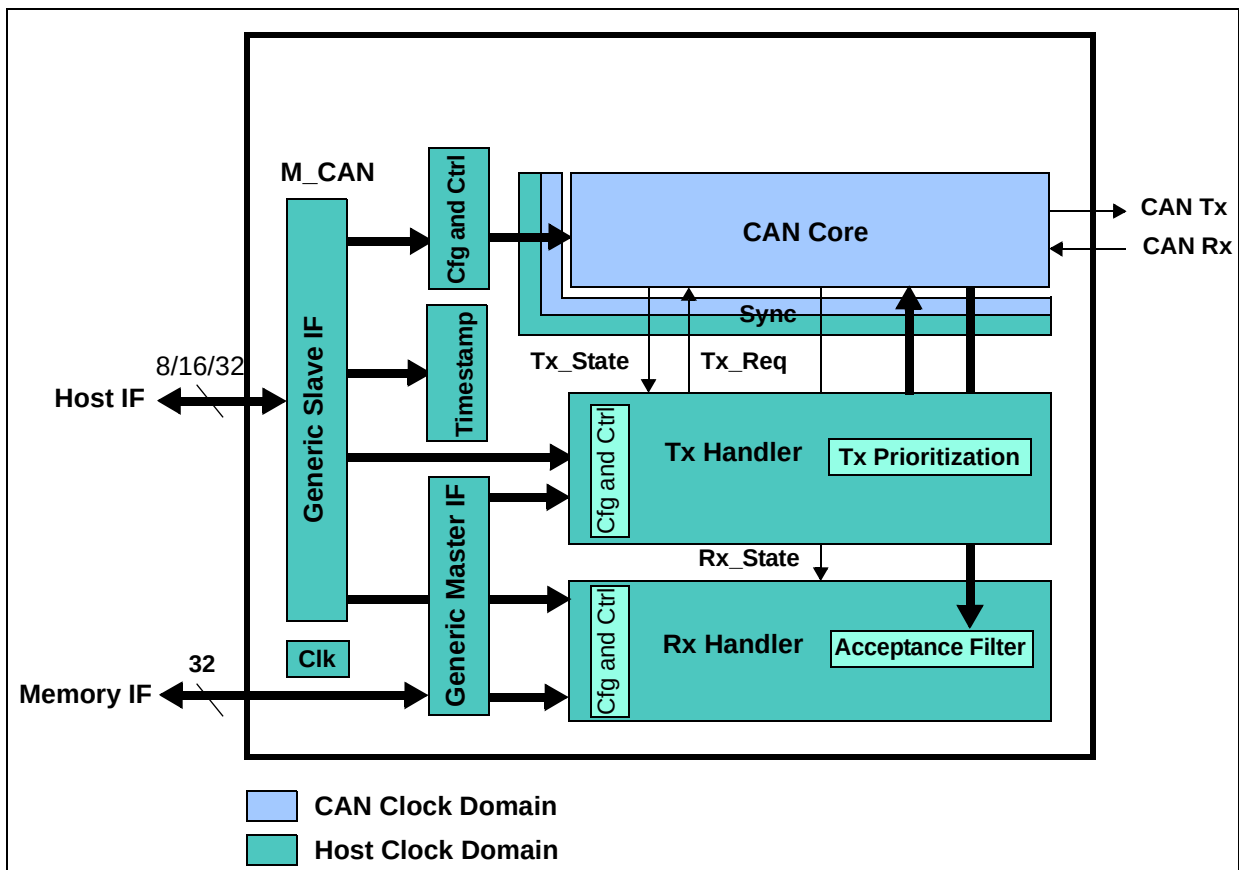


Figure 542. M_CAN core block diagram

- CAN core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Clk: Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Cfg and Ctrl: CAN core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.
- Rx Handler: Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored

together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

- Generic Slave Interface: Connects the M_CAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic Master Interface: Connects the M_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB × 32 bits. A single M_CAN can use at most 1 KB × 32 bits.

46.3.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as CAN clock. The CAN core should be programmed to have at least 8 clocks per bit time, this is for example, 1 Mbaud at CAN clock ≥ 8 MHz. Even in case of a very high Host clock frequency, the clock frequency of the CAN core must not be higher than 8 MHz.

Within the M_CAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note: In order to achieve a stable function of the M_CAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

Caution: Stop this module when the device is in STOP mode. If CAN message reception is mandatory, check that the PLL is unlocked before transmitting any message. Use the external oscillator (XOSC) instead of the internal RC oscillator (IRCOSC) during the STOP mode.

46.3.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to INT 0 or to INT 1. By default all interrupts are routed to interrupt line INT 0. By programming INT 0 and INT 1 bits of the Interrupt Line Enable (ILE) register, the interrupt lines can be enabled or disabled separately.

46.3.5 Memory map and register descriptions

46.3.5.1 Hardware reset description

After hardware reset, the registers of the M_CAN hold the reset values listed in [Table 631](#). Additionally the *Bus_Off* state is reset and the M_CAN Tx output is set to *recessive* (HIGH). The value 0x0001 (**bit 0 [INIT]** of the **CCCR register** = '1') in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets **bit 0 [INIT]** of the **CCCR register** = '0'.

46.3.5.2 Register map

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

Note: Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the bit 1 [CCE] and bit 0 [INIT] of the CCCR register is set to “1”.

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

Table 631. M_CAN memory map

Address Offset	Register Name	Access ⁽¹⁾	Reset value	Section/Page
0x0000	CREL—Core Release Register	R	rrrd dddd	46.3.5.2.1/1380
0x0004	ENDN—Endian Register	R	8765 4321	46.3.5.2.2/1381
0x0008	Reserved	—		
0x000C	FBTP—Fast Bit Timing and Prescaler Register	RP	0x000_0A33	46.3.5.2.3/1381
0x0010	TEST—Test Register	RW	0x0000_0000	46.3.5.2.4/1382
0x0014	RWD—RAM Watchdog Register	RP	0x0000_0000	46.3.5.2.5/1383
0x0018	CCCR—CC Control Register	RP	0x0000_0001	46.3.5.2.6/1385
0x001C	BTP—Bit Timing and Prescaler Register	RP	0x0000_2301	46.3.5.2.7/1387
0x0020	TSCC—Timestamp Counter Configuration Register	RP	0x0000_0000	46.3.5.2.8/1389
0x0024	TSCV—Timestamp Counter Value Register	w1c	0x0000_0000	46.3.5.2.9/1390
0x0028	TOCC—Timeout Counter Configuration Register	RP	0xFFFF_0000	46.3.5.2.10/1391
0x002C	TOCV—Timeout Counter Value Register	w1c	0x0000_FFFF	46.3.5.2.11/1392
0x0030–0x003C	Reserved	—		
0x0040	ECR—Error Counter Register	R	0x0000_0000	46.3.5.2.12/1392
0x0044	PSR—Protocol Status Register	RS	0x0000_0007	46.3.5.2.13/1393
0x0048–0x004C	Reserved	—		
0x0050	IR—Interrupt Register	RW	0x0000_0000	46.3.5.2.14/1395
0x0054	IE—Interrupt Enable Register	RW	0x0000_0000	46.3.5.2.15/1399
0x0058	ILS—Interrupt Line Select Register	RW	0x0000_0000	46.3.5.2.16/1401
0x005C	ILE—Interrupt Line Enable Register	RW	0x0000_0000	46.3.5.2.17/1405
0x0060–0x007C	Reserved	—		
0x0080	GFC—Global Filter Configuration Register	RP	0x0000_0000	46.3.5.2.18/1406
0x0084	SIDFC—Standard ID Filter Configuration Register	RP	0x0000_0000	46.3.5.2.19/1407
0x0088	XIDFC—Extended ID Filter Configuration Register	RP	0x0000_0000	46.3.5.2.20/1408
0x008C	Reserved	—		
0x0090	XIDAM—Extended ID AND Mask Register	RP	0x1FFF_FFFF	46.3.5.2.21/1409

Table 631. M_CAN memory map (continued)

Address Offset	Register Name	Access ⁽¹⁾	Reset value	Section/Page
0x0094	HPMS—High Priority Message Status Register	R	0x0000_0000	46.3.5.2.22/1410
0x0098	NDAT1—New Data 1 Register	RW	0x0000_0000	46.3.5.2.23/1411
0x009C	NDAT2—New Data 2 Register	RW	0x0000_0000	46.3.5.2.24/1411
0x00A0	RXF0C—Rx FIFO 0 Configuration Register	RP	0x0000_0000	46.3.5.2.25/1412
0x00A4	RXF0S—Rx FIFO 0 Status Register	R	0x0000_0000	46.3.5.2.26/1413
0x00A8	RXF0A—Rx FIFO 0 Acknowledge Register	RW	0x0000_0000	46.3.5.2.27/1414
0x00AC	RXBC—Rx Buffer Configuration Register	RW	0x0000_0000	46.3.5.2.28/1415
0x00B0	RXF1C—Rx FIFO 1 Configuration Register	RP	0x0000_0000	46.3.5.2.29/1415
0x00B4	RXF1S—Rx FIFO 1 Status Register	R	0x0000_0000	46.3.5.2.30/1416
0x00B8	RXF1A—Rx FIFO 1 Acknowledge Register	RW	0x0000_0000	46.3.5.2.31/1417
0x00BC	Reserved	—		
0x00C0	TXBC—Tx Buffer Configuration Register	RP	0x0000_0000	46.3.5.2.32/1418
0x00C4	TXFQS—Tx FIFO/Queue Status Register	R	0x0000_0000	46.3.5.2.33/1419
0x00C8	Reserved	—		
0x00CC	TXBRP—Tx Buffer Request Pending Register	R	0x0000_0000	46.3.5.2.34/1420
0x00D0	TXBAR—Tx Buffer Add Request Register	RW	0x0000_0000	46.3.5.2.35/1421
0x00D4	TXBCR—Tx Buffer Cancellation Register	RW	0x0000_0000	46.3.5.2.36/1421
0x00D8	TXBTO—Tx Buffer Transmission Occurred Register	R	0x0000_0000	46.3.5.2.37/1422
0x00DC	TXBCF—Tx Buffer Cancellation Finished Register	R	0x0000_0000	46.3.5.2.38/1423
0x00E0	TXBTIE—Tx Buffer Transmission Interrupt Enable	RW	0x0000_0000	46.3.5.2.39/1423
0x00E4	TXBCIE—Tx Buffer Cancellation Finished Interrupt Enable Register	RW	0x0000_0000	46.3.5.2.40/1424
0x00E8–0x00EC	Reserved	—		
0x00F0	TXEFC—Tx Event FIFO Configuration Register	RP	0x0000_0000	46.3.5.2.41/1425
0x00F4	TXEFS—Tx Event FIFO Status Register	R	0x0000_0000	46.3.5.2.42/1426
0x00F8	TXEFA—Tx Event FIFO Acknowledge Register	RW	0x0000_0000	46.3.5.2.43/1427
0x00FC	Reserved	—		

1. R = Read, S = Set on read, W = Write, P = Protected Write, C = Clear/preset on write, r = release, d = date

46.3.5.2.1 Core Release Register (CREL)

Figure 543. Core Release Register

Address: 0x0000

Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset ⁽¹⁾	r	r	r	r	r	r	r	r	r	r	r	r	d	d	d	d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset ⁽¹⁾	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d

1. R = Read; -r = release, -d = time stamp. The coding of revisions depends on the module version used in the device.

Table 632. Core Release Register Field Descriptions

Field	Description
0:3 REL	Core Release One digit, BCD
4:7 STEP	Step of Core Release One digit, BCD
8:11 SUBSTEP	Sub-step of Core Release One digit, BCD
12:15 YEAR	Time Stamp Year One digit, BCD
16:23 MON	Time Stamp Month Two digits, BCD
24:31 DAY	Time Stamp Day Two digits, BCD

46.3.5.2.2 Endian Register (ENDN)

Figure 544. Endian Register

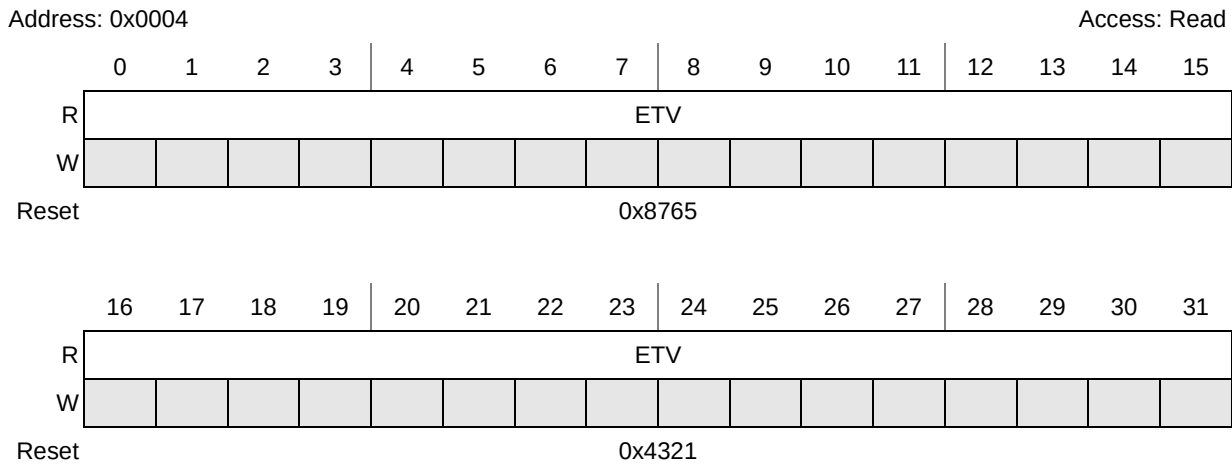


Table 633. Endian Register Field Descriptions

Field	Description
0:31 ETV	Endianness Test Value The endianness test value is 0x87654321.

46.3.5.2.3 Fast Bit Timing and Prescaler Register (FBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 M_CAN clock periods. $tq = (FBRP + 1) M_CAN$ clock period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. **FTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (prog rammed values) **[FTSEG1 + FTSEG2 + 3]** tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Figure 545. Fast Bit Timing and Prescaler Register

Address: 0x000C

Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TDCO ⁽¹⁾				TDC ⁽¹⁾	0	FBRP ⁽¹⁾									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	FTSEG ⁽¹⁾				0	FTSEG2 ⁽¹⁾				0		FSJW ⁽¹⁾	
W													FTSEG2 ⁽¹⁾				
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1	

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Figure 546. Fast Bit Timing and Prescaler Register

Field	Description
0:3 TDCO	Transceiver Delay Compensation Offset 0x0–0xF— Offset value defining the distance between the measured delay from m_can_tx to m_can_rx and the secondary sample point. Valid values are 0 to 15 M_CAN clock periods.
4 TDC	Transceiver Delay Compensation 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
6:15 FBRP	Fast Baud Rate Prescaler 0x000–0x3FF— The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
20:23 FTSEG1	Fast time segment before sample point 0x1–0xF— Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
25:27 FTSEG2	Fast time segment before sample point 0x0–0x7— Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
30:31 FSJW	Fast (Re) Synchronization Jump Width 0x0–0x3— Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 kbit/s.

46.3.5.2.4 Test Register

Write access to the Test Register has to be enabled by setting CCCR[TEST] to '1'. All Test Register functions are set to their reset values when CCCR[TEST] is reset. Loop Back

mode and software control of M_CAN Tx pin are hardware test modes. Programming of TX ≠ "00" may disturb the message transfer on the CAN bus.

Figure 547. Test Register

Address: 0x0010

Access: Read-Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	TDCV					RX	TX ⁽¹⁾			LBCK ⁽¹⁾	0	0	0	0
W																	
Reset	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0	

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 634. Test Register field descriptions

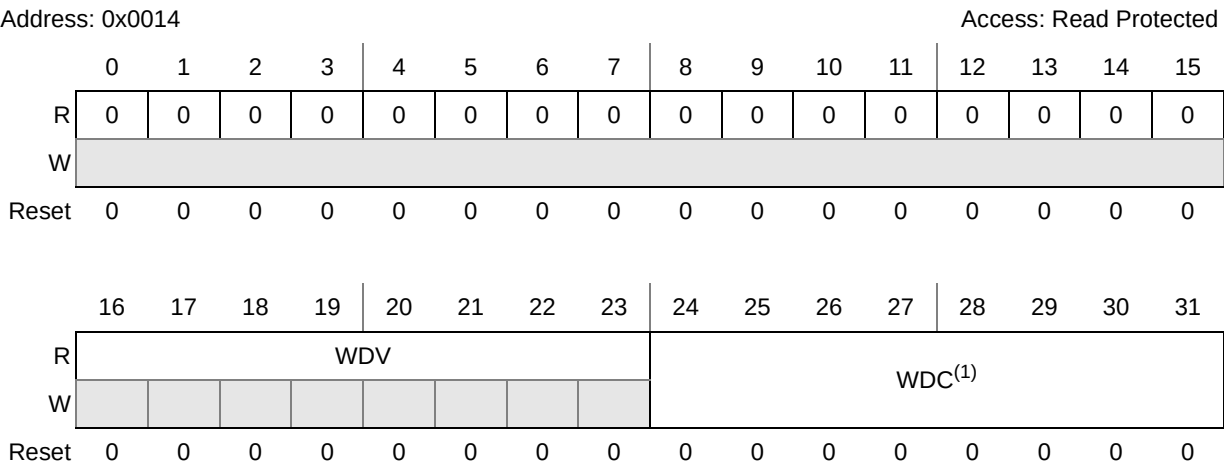
Field	Description
0:14	Reserved
19:23 TDCV	Transceiver Delay Compensation Value 0x00–0x1F— Position of the secondary sample point, defined by the sum of the measured delay from m_can_tx to m_can_rx and FBTP.TDCO. Valid values are 0 to 31 M_CAN core clock periods.
24 RX	Receive Pin Monitors the actual value of pin m_can_rx 0 The CAN bus is dominant (m_can_rx = '0') 1 The CAN bus is recessive (m_can_rx = '1')
25:26 TX	Control of Transmit Pin 00 Reset value, m_can_tx is controlled by the CAN core, updated at the end of the CAN bit time 01 Sample Point can be monitored at pin m_can_tx 10 Dominant ('0') level at pin m_can_tx 11 Recessive ('1') at pin m_can_tx
27 LBCK	Loop Back mode 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (see Section 46.3.7.8, Test modes)
28:31	Reserved

46.3.5.2.5 RAM Watchdog Register (RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the WDC bits. The counter is reloaded with WDC bits when the Message RAM signals successful completion by activating its READY output. In

case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag WDI bit of the IR register is set. The RAM Watchdog Counter is clocked by the Host clock .

Figure 548. RAM watchdog register



1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 635. RAM Watchdog register field descriptions

Field	Description
0:15	Reserved
16:23 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value
24:31 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled.

46.3.5.2.6 CC Control Register (CCCR)

The following section shows the CAN Control register. For details about setting and resetting of single bits see [Section 46.3.7.1.1, Software initialization](#).

Figure 549. CC Control register

Address: 0x0018

Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LONG ⁽¹⁾	FAST ⁽¹⁾	LACT	FACT	CLRL	SETL	CLRF	SETF	TEST ⁽¹⁾	DAR ⁽¹⁾	MON ⁽¹⁾	CSR	CSA	ASM ⁽¹⁾	CCE ⁽¹⁾	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 636. CC Control Register field descriptions

Field	Description
0:15	Reserved
16 LONG	Long Frame Mode Enable 0 Long Frame Mode disabled, LACT may not be set 1 Long Frame Mode enabled, LACT may be set
17 FAST	Fast Frame Mode Enable 0 Fast Frame Mode disabled, FACT may not be set 1 Fast Frame Mode enabled, FACT may be set if Long Frame Mode also enabled
18 LACT	Long Frame Mode Active 0 Frames transmitted in Standard CAN format 1 Frames transmitted in CAN FD Long format
19 FACT	Fast Frame Mode Active 0 Format of transmitted frames defined by LACT 1 Frames transmitted in CAN FD Fast format
20 CLRL	Clear Long Frame Mode 0 Not waiting for deactivation of Long Frame Mode 1 Waiting for deactivation of Long Frame Mode
21 SETL	Set Long Frame Mode 0 Long Frame Mode active or not requested 1 Waiting for activation of Long Frame Mode

Table 636. CC Control Register field descriptions (continued)

Field	Description
22 CLRf	Clear Fast Bit Rate Mode 0 Not waiting for deactivation of Fast Frame Mode 1 Waiting for deactivation of Fast Frame Mode
23 SETf	Set Fast Bit Rate Mode 0 Long Fast Mode active or not requested 1 Waiting for activation of Fast Frame Mode
24 TEST	Test Mode Enable 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	Disable Automatic Retransmission 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_CAN may be set in power down by stopping host clock and core clock
29 ASM	ASM Restricted Operation Mode The Restricted Operation Mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame. In the optional Restricted Operation Mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. If the M_CAN is connected to a Clock Calibration on CAN unit, ASM is controlled by input M_CAN calibration OK (m_can_cok) signal. In case m_can_cok switches to '0', bit ASM is set. When m_can_cok switches back to '1', bit ASM returns to the previously written value. The state of ASM is the written value while input m_can_cok is at '1'. The input is hardwired to '1' when there is no Clock Calibration on CAN unit connected. 0 Normal CAN operation 1 Restricted Operation Mode active

Table 636. CC Control Register field descriptions (continued)

Field	Description
30 CCE	Configuration Change Enable 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = '1')
31 INIT	Initialization 0 Normal Operation 1 Initialization is started

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

46.3.5.2.7 Bit Timing and Prescaler Register (BTP)

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programmed in the range of [4...81] time quanta. The CAN time quantum may be programmed in the range of [1...1024] M_CAN clock periods.

Equation 27 $tq = (BRP + 1) M_CAN \text{ clock period}$

TSEG1 is the sum of Prop_Seg and Phase_Seg1. **TSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) **[TSEG1 + TSEG2 + 3] tq** or (functional values) **[Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq**.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Figure 550. Bit Timing and Prescaler Register

Address: 0x001C												Access: Read Protected				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	BRP ⁽¹⁾									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	TSEG1 ⁽¹⁾						TSEG2[3:0] ⁽¹⁾				SJW ⁽¹⁾			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 637. Bit Timing and Prescaler Field Descriptions

Field	Description
0:5	Reserved
6:15 BRP	Baud Rate Prescaler (0x000–0x3FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16:17	Reserved
18:23 TSEG1	Time segment before sample point (0x01–0x3F) Valid values are 1 to 63. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24:27 TSEG2	The time segment before the sample point (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28:31 SJW	(Re) Synchronization Jump Width (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a bit rate of 500 kbit/s.

Note: The Information Processing Time (IPT) of the M_CAN is 0 tq. Therefore, to conform to the CAN protocol specification, the lengths for phase segment 1 (Phase_Seg1) and phase segment 2 (Phase_Seg2) must be equal.

46.3.5.2.8 Timestamp Counter Configuration Register (TSCC)

Figure 551. Timestamp Counter Configuration Register

Address: 0x0020

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												TCP ⁽¹⁾			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													TSS ⁽¹⁾		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 638. Timestamp Counter Configuration field descriptions

Field	Description
0:11	Reserved
12:15 TCP	Timestamp Counter Prescaler 0x0–0xF Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to 16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. <i>Note: With CAN FD an external counter is required for timestamp generation (TSS = "10")</i>
16:29	Reserved
30:31 TSS	Timestamp Select 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to TCP 10 Reserved 11 Same as "00"

46.3.5.2.9 Timestamp Counter Value Register (TSCV)

Figure 552. Timestamp counter value register

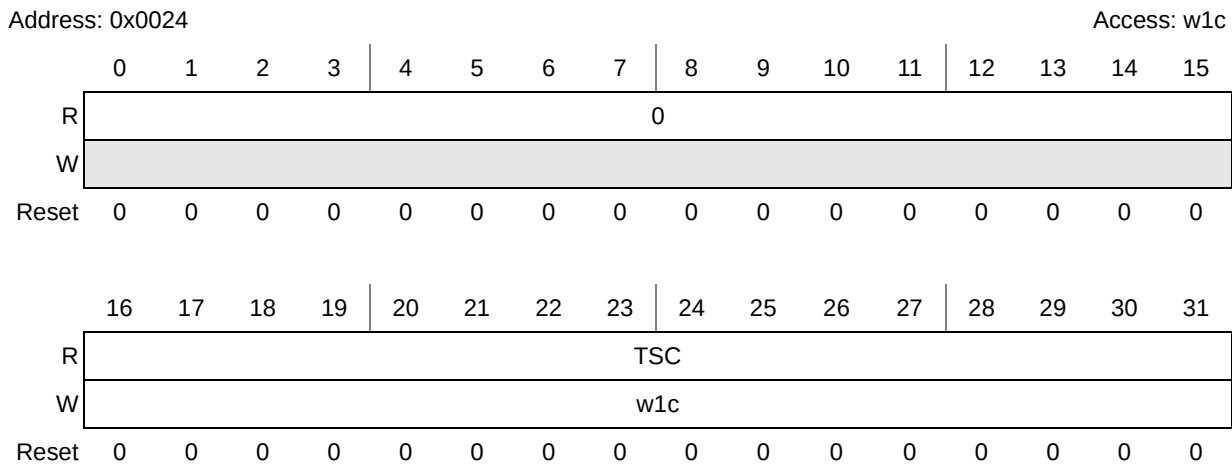


Table 639. Timestamp counter value field descriptions

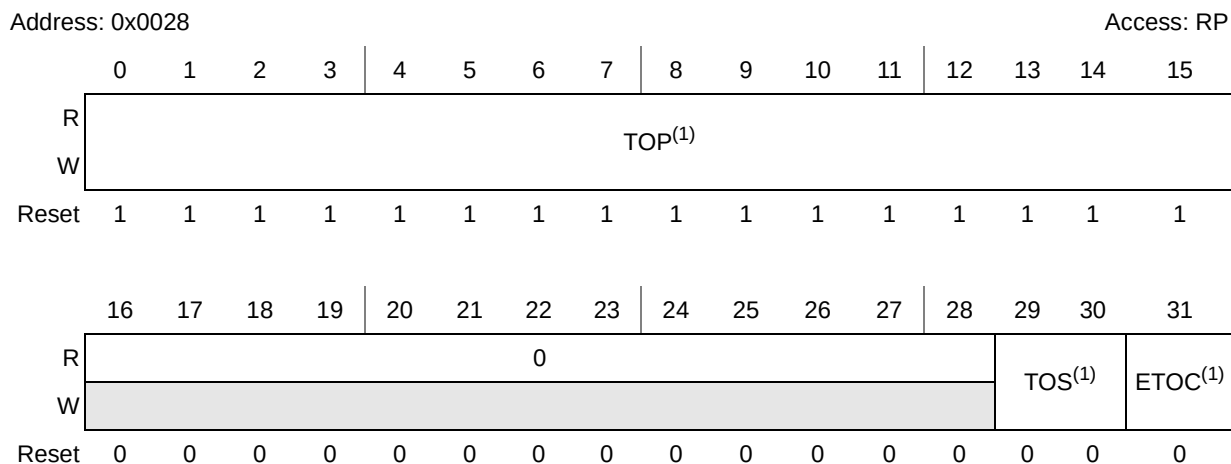
Field	Description
0:15	Reserved
16:30 TSC	Timestamp Counter The Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC[TSS] = "01", the Timestamp Counter is incremented in multiples of CAN bit times [1...16] depending on the configuration of TSCC[TCP] . A wrap around sets interrupt flag IR[TSW] . Write access resets the counter to zero.

Note: A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

46.3.5.2.10 Timeout Counter Configuration Register (TOCC)

For a description of the Timeout Counter see [Section 46.3.9, Timeout counter](#).

Figure 553. Timeout counter configuration register



1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 640. Timeout Counter Configuration Field Descriptions

Field	Description
0:15 TOP[15:0]	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
16:28	Reserved
29:30 TOS	Timeout Select When operating in Continuous mode, a write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP] . Down-counting is started when the first FIFO element is stored. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counter 0 Timeout Counter disabled 1 Timeout Counter enabled <i>Note: For use of timeout function with CAN FD see Section 46.3.9, Timeout counter.</i>

46.3.5.2.11 Timeout Counter Value Register (TOCV)

Figure 554. Timeout counter value register

Address: 0x002C

Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TOC															
W	w1c															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 641. Timeout counter value field descriptions

Field	Description
0:15	Reserved
16:31 TOC	The Timeout Counter is decremented in multiples of CAN bit times [1 to 16] depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

46.3.5.2.12 Error Counter Register (ECR)

Figure 555. Error counter register

Address: 0x0040

Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CEL ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Access type is RX: reset on read.

Table 642. Error counter register field descriptions

Field	Description
0:7	Reserved
8:15 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag ELO bit of the IR register.
16 RP	Receive Error Passive 0 The Receive Error Counter is below the error passive level of 128 1 The Receive Error Counter has reached the error passive level of 128
17:23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127
24:31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255

Note: When *CCCR.ASM* is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

46.3.5.2.13 Protocol Status Register (PSR)

Figure 556. Protocol status register (PSR)

Address: 0x0044

Access: RS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	REDL (1)	RBR (1)	RESI (1)	FLEC ⁽²⁾			BO	EW	EP	ACT		LEC ⁽²⁾		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

1. Access type is RX: reset on read.

2. Access type is RS: set on read.

Table 643. PSR descriptions

Field	Description
0:17	Reserved
18 REDL	Received CAN FD Message with EDL flag 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with EDL flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not have its BRS flag set. 1 Last received CAN FD message had its BRS flag set.
20 RESI	ESI CAN FD Message with ESI flag This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not have its ESI flag set. 1 Last received CAN FD message had its ESI flag set
21:23 FLEC	Fast Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.
24 BO	Bus_Off Status 0 The M_CAN is not Bus_Off 1 The M_CAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_CAN is in the Error_Passive state
27:28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing – node is synchronizing on CAN communication 01 Idle—node is neither receiver nor transmitter 10 Receiver—node is operating as receiver 11 Transmitter—node is operating as transmitter

Table 643. PSR descriptions (continued)

Field	Description
29:31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_CAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCErrror: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT]. If the device goes Bus_Off, it will set CCCR[INIT] of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129×11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit 0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

46.3.5.2.14 Interrupt Register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Figure 557. Interrupt register

Address: 0x0050

Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STE	FOE	ACKE	BE	CRCE	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	UMD	TSW
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 644. Interrupt register field descriptions

Field	Description
0 STE	Stuff Error 0 No Stuff Error detected 1 More than 5 equal bits in a sequence occurred
1 FOE	Format Error 0 No Format Error detected 1 A fixed format part of a received frame has the wrong format
2 ACKE	Acknowledge Error 0 No Acknowledge Error detected 1 A transmitted message was not acknowledged by another node
3 BE	Bit Error 0 No Bit Error detected 1 Device wanted to send a rec / dom level, but monitored bus level was dom / rec
4 CRCE	CRC Error 0 No CRC Error detected 1 Received CRC did not match the calculated CRC
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed

Table 644. Interrupt register field descriptions (continued)

Field	Description
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to '1'. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (e.g. parity logic)
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (e.g. ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 UMD	Unprocessed Message Discarded When a new message is received while the acceptance filtering process for the previously received message has not yet completed, this message is discarded. 0 No unprocessed message discarded 1 Unprocessed message discarded
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Event Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark

Table 644. Interrupt register field descriptions (continued)

Field	Description
19 TEFN	Tx Event FIDO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

46.3.5.2.15 Interrupt Enable Register (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt register will be signaled on an interrupt line.

Figure 558. Interrupt enable register

Address: 0x0054

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	UMDE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLE	TEFFE	TEFWE	TEFNE	TTEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 645. Interrupt enable register field descriptions

Field	Description
0 STEE	Stuff Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
1 FOEE	Format Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
2 ACKEE	Acknowledge Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
3 BEE	Bit Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
4 CRCEE	CRC Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 645. Interrupt enable register field descriptions (continued)

Field	Description
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 UMDE	Unprocessed Message Discarded Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 645. Interrupt enable register field descriptions (continued)

Field	Description
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

46.3.5.2.16 Interrupt Line Select Register (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

Figure 559. Interrupt line select register

Address: 0x0058

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	UMDL	TSWL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 646. Interrupt line select register field descriptions

Field	Description
0 STEL	Stuff Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
1 FOEL	Format Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
2 ACKEL	Acknowledge Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
3 BEL	Bit Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
4 CRCEL	CRC Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 646. Interrupt line select register field descriptions (continued)

Field	Description
9 EOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
12 UMDL	UMDL: Unprocessed Message Discarded Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
16 TEFL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 646. Interrupt line select register field descriptions (continued)

Field	Description
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

46.3.5.2.17 Interrupt Line Enable Register (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

Figure 560. Interrupt line enable register

Address: 0x005C																Access: RW	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W																EINT1	EINT0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 647. Interrupt line enable register field descriptions

Field	Description
0:29 STEL	Reserved
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

46.3.5.2.18 Global Filter Configuration Register (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in [Figure 595](#) and [Figure 596](#).

Figure 561. Global filter configuration register

Address: 0x0080

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ANFS ⁽¹⁾		ANFE ⁽¹⁾		RRFS ⁽¹⁾	RRFE ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 648. Global filter configuration register field descriptions

Field	Description
0:25	Reserved
26:27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28:29 ANFE	ANFE[1:0]: Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

46.3.5.2.19 Standard ID Filter Configuration Register (SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in [Figure 589](#).

Figure 562. Standard ID Filter Configuration register

Address: 0x0084												Access: RP				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LSS ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLSSA[15:2] ⁽¹⁾														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 649. Standard ID Filter Configuration register field descriptions

Field	Description
0:7 STEL	Reserved
8:15 LSS	List Size Standard 0 No standard Message ID filter 1–128 Number of standard Message ID filter elements >128 Values greater than 128 are interpreted as 128
16:29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, see Figure 585).
30:31	Reserved

46.3.5.2.20 Extended ID Filter Configuration Register (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in [Figure 590](#).

Figure 563. Extended ID Filter Configuration register

Address: 0x0088														Access: RP			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0	0	0	0	0	0	0	0	0	LSE ⁽¹⁾							
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLESA ⁽¹⁾														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

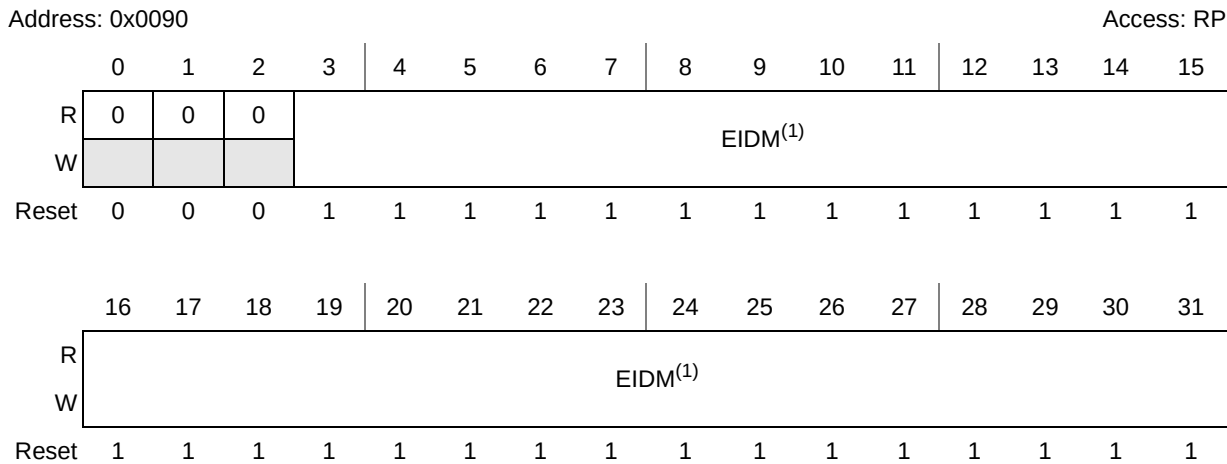
1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 650. Extended ID Filter Configuration register field descriptions

Field	Description
0:8	Reserved
9:15 LSE	List Size Extended 0 No extended Message ID filter 1–64 Number of extended Message ID filter elements >64 Values greater than 64 are interpreted as 64
16:29 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, see Figure 585).
30:31	Reserved

46.3.5.2.21 Extended ID and Mask Register (XIDAM)

Figure 564. Extended ID and Mask register



1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 651. Extended ID and Mask register field descriptions

Field	Description
0:2	Reserved
3:31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

46.3.5.2.22 High Priority Message Status Register (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Figure 565. High Priority Message Status register

Address: 0x0094

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	FLST	FIDX								MSI		BIDX					
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 652. High Priority Message Status register field descriptions

Field	Description
0:15	Reserved
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17:23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24:25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26:31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = '1'.

46.3.5.2.23 New Data 1 Register (NDAT1)

Figure 566. NDAT1 register

Address: 0x0098

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 567. NDAT1 register field descriptions

Field	Description
0 ND	New Data[31:0] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

46.3.5.2.24 New Data 2 Register (NDAT2)

Figure 568. NDAT2 register

Address: 0x009C

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32	ND31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 569. NDAT2 register field descriptions

Field	Description
0 ND	New Data[63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

46.3.5.2.25 Rx FIFO 0 Configuration Register (RXF0C)

Figure 570. Rx FIFO 0 Configuration register

Address: 0x00A0

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F0WM ⁽¹⁾							0	F0S ⁽¹⁾						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F0SA ¹														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 653. Rx FIFO 0 Configuration register field descriptions

Field	Description
0	Reserved
1:7 F0WM	Rx FIFO 0 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) >64 Watermark interrupt disabled
8:9	Reserved
10:15 F0S	Rx FIFO 0 Size 0 No Rx FIFO 0 1-64 Number of Rx FIFO 0 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1

Table 653. Rx FIFO 0 Configuration register field descriptions (continued)

Field	Description
16:29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, Figure 585).
30:31	Reserved

46.3.5.2.26 Rx FIFO 0 Status Register (RXF0S)

Table 654. Rx FIFO 0 Status Register

Address: 0x00A4

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RF0L	F0F	0	0	F0PI					
W																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F0GI						0	F0FL						
W																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 655. Rx FIFO 0 Status register field descriptions

Field	Description
0:5	Reserved
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L] . When IR[RF0L] is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
8:9	Reserved
10:15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.
16:17	Reserved
18:23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.

Table 655. Rx FIFO 0 Status register field descriptions (continued)

Field	Description
24	Reserved
25:31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

46.3.5.2.27 Rx FIFO 0 Acknowledge Register (RXF0A)

Figure 571. Rx FIFO 0 Acknowledge register

Address: 0x00A8

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F0AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 656. Rx FIFO 0 Acknowledge register field descriptions

Field	Description
0:25	Reserved
26:31 F0AI	Rx FIFO 0 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI . This will set the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL] .

46.3.5.2.28 Rx Buffer Configuration Register (RXBC)

Figure 572. Rx FIFO 0 Configuration register

Address: 0x00AC

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RBSA ¹														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 657. Rx FIFO 0 Configuration register field descriptions

Field	Description
0:15	Reserved
16:29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.
30:31	Reserved

46.3.5.2.29 Rx FIFO 1 Configuration Register (RXF1C)

Figure 573. Rx FIFO 0 Configuration register

Address: 0x00B0

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F1WM ⁽¹⁾						0	F1S							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F1SA ⁽¹⁾														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 658. Rx FIFO 0 Configuration register field descriptions

Field	Description
0	Reserved
1:7 F1WM	Rx FIFO 1 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 1 watermark interrupt IR[RF1W] >64 Watermark interrupt disabled
8	Reserved
9:15 F1S	F1S[6:0]: Rx FIFO 1 Size 0 No Rx FIFO 1 1-64 Number of Rx FIFO 1 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1
16:29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, see Figure 585).
30:31	Reserved

46.3.5.2.30 Rx FIFO 1 Status Register (RXF1S)

Figure 574. Rx FIFO 1 Status register

Address: 0x00B4

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS	0	0	0	0	RF1L	F1F	0	0	F1PI						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 659. Rx FIFO 1 Status register field descriptions

Field	Description
0:1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
2:5	Reserved

Table 659. Rx FIFO 1 Status register field descriptions (continued)

Field	Description
6 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag IR[RF1L] . When IR[RF1L] is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
8:9	Reserved
10:15 F1P1	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
16:17	Reserved
18:23 F1G1	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
24	Reserved
25:31 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

46.3.5.2.31 Rx FIFO 1 Acknowledge Register (RXF1A)

Figure 575. Rx FIFO 1 Acknowledge register

Address: 0x00B8

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F1AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 660. Rx FIFO 1 Acknowledge register field descriptions

Field	Description
0:25	Reserved
26:31 F1AI	Rx FIFO 1 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI . This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL] .

46.3.5.2.32 Tx Buffer Configuration Register (TXBC)

Figure 576. Tx Buffer Configuration register

Address: 0x00C0

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TFQM ⁽¹⁾	TFQS ⁽¹⁾						0	0	NDTB ⁽¹⁾					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBSA ⁽¹⁾														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 661. Tx Buffer Configuration register field descriptions

Field	Description
0	Reserved
1 TQFM	Tx FIFO/Queue Mode 0 Tx FIFO operation 1 Tx Queue operation
2:7 TQFS	Tx FIFO/Queue Size 0 No Tx FIFO/Queue 1-32 Number of Tx Buffers used for Tx FIFO/Queue >32 Values greater than 32 are interpreted as 32
8:9	Reserved
10:15 NDTB	Number of Dedicated Transmit Buffers 0 No Dedicated Tx Buffers 1-32 Number of Dedicated Tx Buffers >32 Values greater than 32 are interpreted as 32

Table 661. Tx Buffer Configuration register field descriptions (continued)

Field	Description
16:29 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, see Figure 585).
30:31	Reserved

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

46.3.5.2.33 Tx FIFO/Queue Status Register (TXFQS)

Figure 577. Tx FIFO/Queue Status register

Address: 0x00C4

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TFGI					0	0	TFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 662. Tx Buffer Configuration register field descriptions

Field	Description
0:9	Reserved
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11:15 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31.
16:18	Reserved
19:23 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = '1').
24:25	Reserved
26:31 TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = '1')

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

46.3.5.2.34 Tx Buffer Request Pending Register (TXBRP)

Table 663. Tx Buffer Request Pending register

Address: 0x00CC

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 664. Tx Buffer Request Pending register field descriptions

Field	Description
0:31 TRP	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (see Section 46.4.7.15, Tx handling) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF – after successful transmission together with the corresponding TXBTO bit – when the transmission has not yet been started at the point of cancellation – when the transmission has been aborted due to lost arbitration – when an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

Note: *TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.*

46.3.5.2.35 Tx Buffer Add Request Register (TXBAR)

Table 665. Tx Buffer Add Request register

Address: 0x00D0

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 666. Tx Buffer Add Request register field descriptions

Field	Description
0:31 AR	Add Request Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

46.3.5.2.36 Tx Buffer Cancellation Request Register (TXBCR)

Table 667. Tx Buffer Cancellation Request register (TXBCR)

Address: 0x00D4

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CR 31	CR 30	CR 29	CR 28	CR 27	CR 26	CR 25	CR 24	CR 23	CR 22	CR 21	CR 20	CR 19	CR 18	CR 17	CR 16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CR 15	CR 14	CR 13	CR 12	CR 11	CR 10	CR 9	CR 8	CR 7	CR 6	CR 5	CR 4	CR 3	CR 2	CR 1	CR 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 668. TXBCR field descriptions

Field	Description
0:31 CR	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

46.3.5.2.37 Tx Buffer Transmission Occurred Register (TXBTO)

Table 0-1.

Address: 0x00D8

Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 578. Tx Buffer Transmission Occurred register

Table 669. Tx Buffer Transmission Occurred field description

Field	Description
[0:31] TO	Transmission Occurred. Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

46.3.5.2.38 Tx Buffer Cancellation Finished Register (TXBCF)

Address: 0x00DC

Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CF															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CF															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 579. Tx Buffer Cancellation Finished register

Table 670. Tx Buffer Cancellation Finished field description

Field	Description
[0:31] CF	Cancellation Finished. Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

46.3.5.2.39 Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Address: 0x00E0

Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 580. Tx Buffer Transmission Interrupt Enable register

Table 671. Tx Buffer Transmission Interrupt field description

Field	Description
[0:31] TIE	Transmission Interrupt Enable. Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

46.3.5.2.40 Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)**Figure 581. Tx Buffer Cancellation Finished Interrupt Enable register**

Address: 0x00E4

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 672. Tx Buffer Cancellation Finished Interrupt Enable field description

Field	Description
[0:31] CFIE	Cancellation Finished Interrupt Enable. Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

46.3.5.2.41 Tx Event FIFO Configuration Register (TXEFC)

Address: 0x00F0

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Figure 582. Tx Event FIFO Configuration register

Table 673. Tx Event FIFO Configuration field description

Field	Description
[0:1]	Reserved.
[2:7] EFWM	Event FIFO Watermark. 0 Watermark interrupt disabled 1–32 Level for Tx Event FIFO watermark interrupt (IR[TEFW]) >32 Watermark interrupt disabled
[8:9]	Reserved.
[10:15] EFS	Event FIFO Size. 0 Tx Event FIFO disabled 1–32 Number of Tx Event FIFO elements >32 Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS - 1
[16:29] EFSA	Event FIFO Start Address. Start address of Tx Event FIFO in Message RAM (32-bit word address, Figure 585 .)
[30:31]	Reserved.

46.3.5.2.42 Tx Event FIFO Status Register (TXEFS)

Address: 0x00F4

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					TEFL	EFF		0				EFPI			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				EFGI				0			EFFL				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 583. Tx Event FIFO Status register

Table 674. Tx Event FIFO Status field description

Field	Description
[0:5]	Reserved.
6 TEFL	Tx Event FIFO Element Lost. This bit is a copy of interrupt flag IR[TEFL] . When IR[TEFL] is reset, this bit is also reset. 0No Tx Event FIFO element lost 1Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
7 EFF	Event FIFO Full. 0Tx Event FIFO not full 1Tx Event FIFO full
[8:10]	Reserved.
[11:15] EFPI	Event FIFO Put Index. Tx Event FIFO write index pointer, range 0 to 31.
[16:18]	Reserved
[19:23] EFGI	Event FIFO Get Index. Tx Event FIFO read index pointer, range 0 to 31.
[24:25]	Reserved.
[26:31] EFFL	Event FIFO Fill Level. Number of elements stored in Tx Event FIFO, range 0 to 32.

46.3.5.2.43 Tx Event FIFO Acknowledge Register (TXEFA)

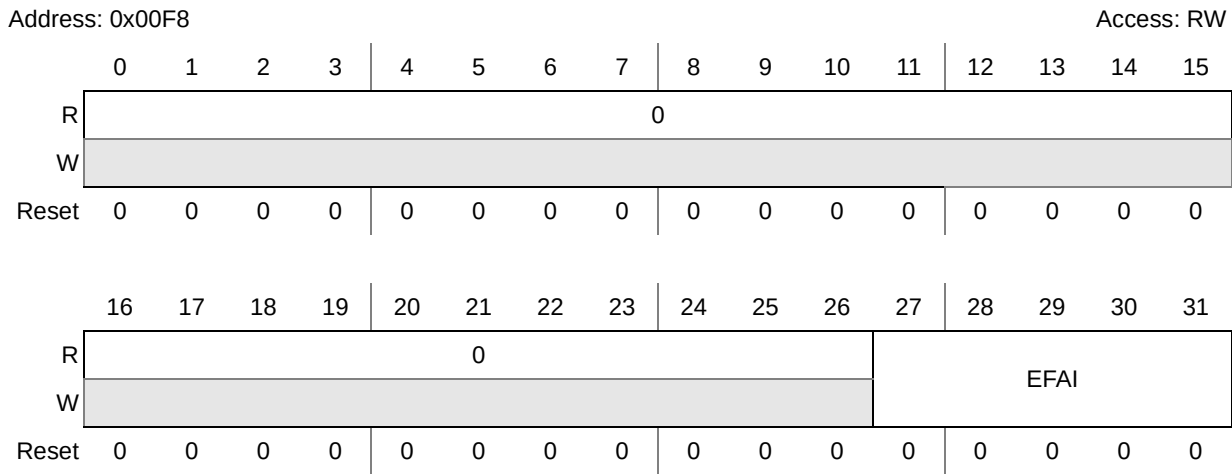


Figure 584. Tx Event FIFO Acknowledge register

Table 675. Tx Event FIFO Acknowledge field description

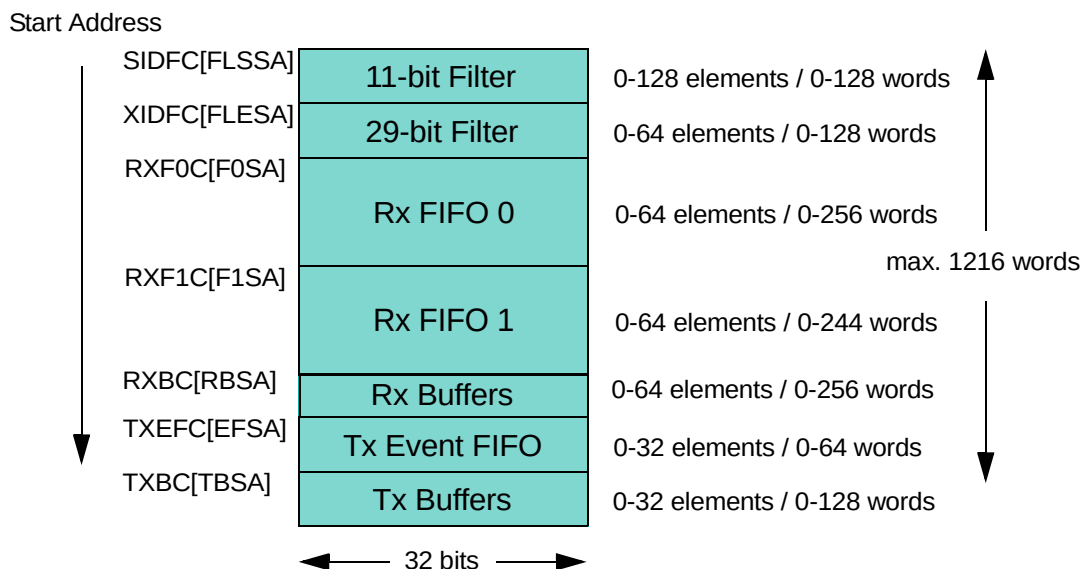
Field	Description
[0:26]	Reserved.
[27:31] EFAI	Event FIFO Acknowledge Index. After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL] .

46.3.6 Message RAM

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 1216 words in the Message RAM. It is not necessary to

configure each of the sections listed in the following figure, nor is there any restriction with respect to the sequence of the sections.

Figure 585. Message RAM Configuration



When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses.

Note: The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

46.3.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the following figure.

Figure 586. Rx FIFO Element

	31	24			23	16			15	8		7	0
R0	ESI	XTD	RTR	ID[28:0]									
R1	ANMF	FIDX[6:0]			res	EDL	BRS	DLC[3:0]	RXTS[15:0]				
R2	DB3[7:0]				DB2[7:0]				DB1[7:0]			DB0[7:0]	
R3	DB7[7:0]				DB6[7:0]				DB5[7:0]			DB4[7:0]	

Table 676. Rx FIFO Element descriptions

R0 Bit 31 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 Bit 30 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit extended identifier
R0 Bit 29 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame
R0 Bits 28:0 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 Bit 21 EDL Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1 Bit 20 BRS Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]: Data Length Code	0-8 Received frame has 0-8 data bytes 9-15 Received frame has 8 data bytes
R1 Bits 15:0 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 Bits 31:24	DB3[7:0]: Data Byte 3
R2 Bits 23:16	DB2[7:0]: Data Byte 2
R2 Bits 15:8	DB1[7:0]: Data Byte 1
R2 Bits 7:0	DB0[7:0]: Data Byte 0
R3 Bits 31:24	DB7[7:0]: Data Byte 7
R3 Bits 23:16	DB6[7:0]: Data Byte 6
R3 Bits 15:8	DB5[7:0]: Data Byte 5
R3 Bits 7:0	DB4[7:0]: Data Byte 4

46.3.6.2 Tx buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC[TFQS] and TXBC[NDTB].

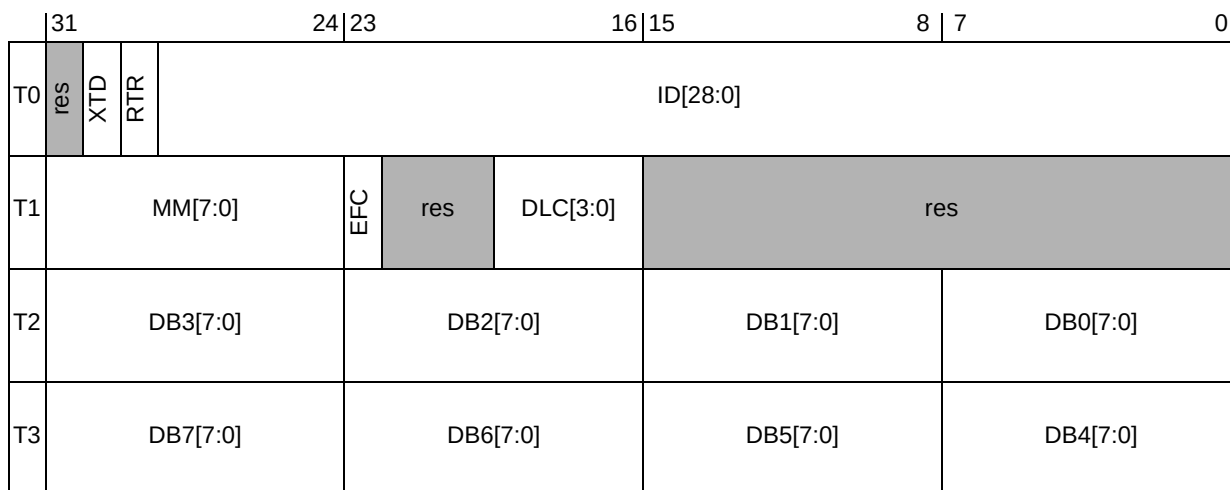


Figure 587. Tx Buffer Element

Table 677. Tx Buffer Element descriptions

T0 Bit 30 XTD: Extended Identifier	0 11-bit standard identifier 1 29-bit extended identifier
T0 Bit 29 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame
T0 Bits 28:0 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO Control 0 Don't store Tx events 1 Store Tx events
T1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes
T2 Bits 31:24	DB3[7:0]: Data Byte 3
T2 Bits 23:16	DB2[7:0]: Data Byte 2

Table 677. Tx Buffer Element descriptions (continued)

T2 Bits 15:8	DB1[7:0]: Data Byte 1
T2 Bits 7:0	DB0[7:0]: Data Byte 0
T3 Bits 31:24	DB7[7:0]: Data Byte 7
T3 Bits 23:16	DB6[7:0]: Data Byte 6
T3 Bits 15:8	DB5[7:0]: Data Byte 5
RT3 Bits 7:0	DB4[7:0]: Data Byte 4

46.3.6.3 Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

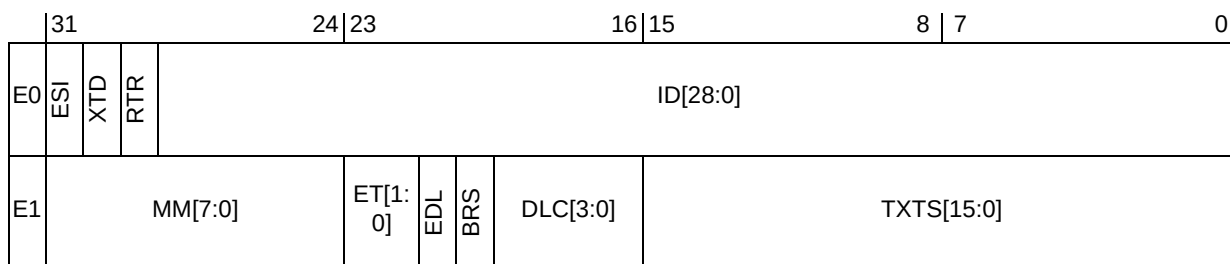


Figure 588. Tx Event FIFO Element

Table 678. Tx Event FIFO Element

E0 Bit 31 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 Bit 30 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 Bit 29 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 Bits 31:24 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status

Table 678. Tx Event FIFO Element (continued)

E1 Bit 23:22 ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 Bit 21 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching
E1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 Frame with 0-8 data bytes transmitted 9-15 Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

46.3.6.4 Standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC[FLSSA] plus the index of the filter element (0 to 127).

	31		24	23		16	15		8	7		0
S0	SFT[1:0]	SFEC[2:0]	SFID1[10:0]			res			SFID2[10:0]			

Figure 589. Standard Message ID Filter Element

Table 679. Standard Message ID Filter Element Field Description

Bits 31:30 SFT[1:0]	Standard Filter Type 00 Range filter from SF1ID to SF2ID ($SF2ID \geq SF1ID$) 01 Dual ID filter for SF1ID or SF2ID 10 Classic filter: SF1ID = filter, SF2ID = mask 11 Reserved
Bit 29:27 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = "100", "101", or "110" a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored
Bits 26:16 SFID1[10:0]	Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 10:0 SFID2[10:0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: 1) SFEC = "001"... "110" Second ID of standard ID filter element 2) SFEC = "111" Filter for Rx Buffers or for debug messages SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C SFID2[8:6] is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_CAN host clock period in case the filter matches. SFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

Note: In case a reserved value is configured, the filter element is considered disabled.

46.3.6.5 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address $XIDFC[FLESA]$ plus two times the index of the filter element (0...63).

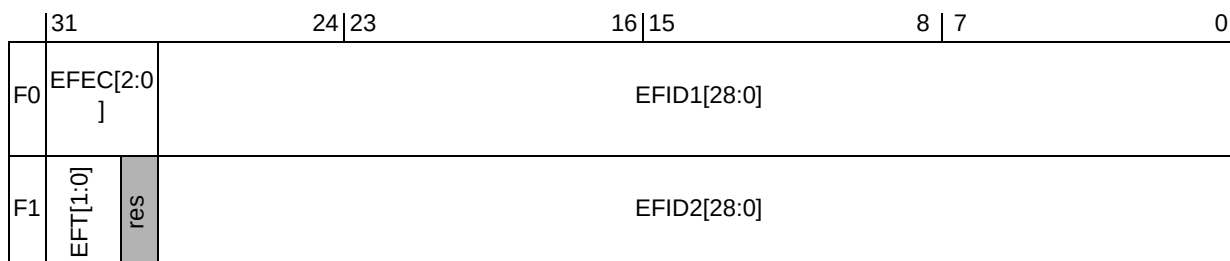


Figure 590. Extended Message ID Filter Element

Table 680. Extended Message ID Filter Element Field Description

F0 Bits 31:29 EFEC[2:0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = "100", "101", or "110" a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored
F0 Bits 28:0 EFID1[28:0]	Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (see Section 46.3.6.5, Extended message ID filter element) is used.

Table 680. Extended Message ID Filter Element Field Description (continued)

F1 Bits 31:30 EFT[1:0]	Extended Filter Type 00 Range filter from EF1ID to EF2ID ($EF2ID \geq EF1ID$) 01 Dual ID filter for EF1ID or EF2ID 10 Classic filter: EF1ID = filter, EF2ID = mask 11 Range filter from EF1ID to EF2ID ($EF2ID \geq EF1ID$), XIDAM mask not applied
F1 Bits 28:0 EFID2[28:0]	This bit field has a different meaning depending on the configuration of EFEC: 1) EFEC = "001"... "110" Second ID of extended ID filter element 2) EFEC = "111" Filter for Rx Buffers or for debug messages EFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C EFID2[8:6] is used to control the filter event pins m_can_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_can_hclk period in case the filter matches. EFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

Note: In case a reserved value is configured, the filter element is considered disabled.

46.3.7 M_CAN functional description

46.3.7.1 Operating modes

46.3.7.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus transmit output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = '1'. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set

- HPMS—High Priority Message Status
- RXF0S—Rx FIFO 0 Status
- RXF1S—Rx FIFO 1 Status
- TXFQS—Tx FIFO/Queue Status
- TXBRP—Tx Buffer Request Pending
- TXBTO—Tx Buffer Transmission Occurred
- TXBCF—Tx Buffer Cancellation Finished
- TXEFS—Tx Event FIFO Status

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOC] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = '1'.

The following registers are only writable while CCCR[CCE] = '0'

- TXBAR - Tx Buffer Add Request
- TXBCR - Tx Buffer Cancellation Request

CCCR[TEST] and CCCR.MON can only be set by the Host while CCCR[INIT] = '1' and CCCR[CCE] = '1'. Both bits may be reset at any time. CCCR.DAR can only be set/reset while CCCR[INIT] = '1' and CCCR[CCE] = '1'.

Following sequence can be used as an example code for initializing the CAN module (M_CAN0 in example):

```
-- Setting the INIT bit high to start the initialization
    while (read_data != 0x003){
        write(M_CAN0, 0xF, driver.regs.can0_cccr.addr, 0x003);
        read_data = read(M_CAN0, 0xF, driver.regs.can0_cccr.addr);
    };

-- Configuring the time stamp and timeout registers
    write(M_CAN0, 0xF, driver.regs.can0_tscc.addr, 0x000);
    write(M_CAN0, 0xF, driver.regs.can0_tocc.addr, 0x000);

-- Configuring the interrupt registers
    write(M_CAN0, 0xF, driver.regs.can0_ie.addr, 0xFFFFFFFF);
    write(M_CAN0, 0xF, driver.regs.can0_ils.addr, 0x00000002);
    write(M_CAN0, 0xF, driver.regs.can0_ile.addr, 0x00000003);

    write(M_CAN0, 0xF, driver.regs.can0_gfc.addr, 0x000);
    write(M_CAN0, 0xF, driver.regs.can0_xidam.addr, 0x00FFFFFF);

--Writing the start addresses of all the M_CAN Buffers
    generate and drive M_CAN0 can0_rxf0c keeping {
                                                .f0sa == 0x100;
                                                .f0s  == 0x03;
```

```

        .f0wm == 0x01;
    };

    generate and drive M_CAN0 can0_rxflc keeping {
        .flsa == 0x200;
        .fls  == 0x03;
        .flwm == 0x01;
    };

    generate and drive M_CAN0 can0_sidfc keeping {
        .flssa == 0x000;
        .lss   == 0x80;
    };

    generate and drive M_CAN0 can0_xidfc keeping {
        .flesa == 0x080;
        .lse   == 0x40;
    };

    generate and drive M_CAN0 can0_txbc  keeping {
        .tbasa == 0x300;
        .ndtb  == 0x020;
        .tfqs  == 0x1D;
    };

    generate and drive M_CAN0 can0_txefc keeping {
        .efsa == 0x380;
        .efs  == 0x020;
        .efwm == 0x00;
    };

--Writing to the RAM for filter configuration
=====
--    Writing to TX Buffer
    write(RAM, 0xF, 0xC00, 0x04040000);
    write(RAM, 0xF, 0xC04, 0x10830000);
    write(RAM, 0xF, 0xC08, 0x00010001);
    write(RAM, 0xF, 0xC10, 0x50000002);
    write(RAM, 0xF, 0xC14, 0x11830000);
    write(RAM, 0xF, 0xC18, 0x00070102);

--    Writing to s-filter
    write(RAM, 0xF, 0x000, 0x58020004);
    write(RAM, 0xF, 0x004, 0x08000005);
    write(RAM, 0xF, 0x008, 0x58060007);
    write(RAM, 0xF, 0x00C, 0x08030008);
    write(RAM, 0xF, 0x010, 0x880E07FB);
    write(RAM, 0xF, 0x014, 0x980D07FE);
    write(RAM, 0xF, 0x018, 0x68100100);
    write(RAM, 0xF, 0x01C, 0x70120200);
    write(RAM, 0xF, 0x020, 0x981F07F6);

```

```

        write(RAM, 0xF, 0x024, 0x50140014);
        write(RAM, 0xF, 0x028, 0x58160017);
        write(RAM, 0xF, 0x02C, 0x08110019);
        write(RAM, 0xF, 0x030, 0x4A22001A);
        write(RAM, 0xF, 0x034, 0xA01A07FF);
        write(RAM, 0xF, 0x038, 0x5BFF07E3);
        write(RAM, 0xF, 0x03C, 0x080007FF);

--   Writing to x-filter
        write(RAM, 0xF, 0x200, 0x201ABCD9);
        write(RAM, 0xF, 0x204, 0x001ABCE5);
        write(RAM, 0xF, 0x208, 0x403BCDEF);
        write(RAM, 0xF, 0x20C, 0x003BCDEF);
        write(RAM, 0xF, 0x210, 0x605CDEF2);
        write(RAM, 0xF, 0x214, 0x405CDEF0);
        write(RAM, 0xF, 0x218, 0x407DEEFF);
        write(RAM, 0xF, 0x21C, 0x007DEF01);
        write(RAM, 0xF, 0x220, 0x209ABCDE);
        write(RAM, 0xF, 0x224, 0x009ABCDF);
        write(RAM, 0xF, 0x228, 0xA0BBCDEF);
        write(RAM, 0xF, 0x22C, 0x40BBCDEE);
        write(RAM, 0xF, 0x230, 0x20DCDEF0);
        write(RAM, 0xF, 0x234, 0x9FFFFFFF);
        write(RAM, 0xF, 0x238, 0x40000000);
        write(RAM, 0xF, 0x23C, 0x1FFFFFFF);

=====
write(M_CAN0, 0xF, driver->regs.can0_btp.addr, 0x00002303);

        print "Configuration done";
        read_data = 0x01;
        while (read_data != 0x00){
            write(M_CAN0, 0xF, driver->regs.can0_cccr.addr, 0x000);
        wait [1];
        read_data = read(M_CAN0, 0xF, driver->regs.can0_cccr.addr);
        };

        print "Initialization done";

```

46.3.7.2 Normal operation

Once the M_CAN is initialized and CCCR[INIT] is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

46.3.7.3 CAN FD Operation

There are two stages in the CAN FD protocol, first the Long Frame Mode where the data field of a CAN frame may be longer than 8 bytes. The second stage is the Fast Frame Mode where control field, data field, and CRC field of a CAN Frame are transmitted with a higher bit rate than the beginning and the end of the frame. Fast Frame Mode can only be used in combination with Long Frame Mode.

The CAN operation mode is chosen by programming CCCR.CME. In case CCCR.CME = g01h transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CCCR.CME = g10h/g11h transmission and reception of long and fast CAN FD frames is enabled. CCCR.CME can only be changed while CCCR.INIT and CCCR.CCE are both set.

When initialization is left (CCCR.INIT set to e0f), both Long Frame Mode and Fast Frame Mode are inactive, they have to be requested by writing to CCCR.CMR. A mode change requested by writing to CCCR.CMR will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event CCCR.CMR is reset to g00h and the status flags CCCR.FACT and CCCR.LACT are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CCCR.CMR is retained until it is overwritten by the next mode change request. Default is normal CAN operation. A change of the CAN operation mode should not be requested while there are pending transmission requests.

When CCCR.CME = g00h, received frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD long frame, EDL = dominant signifies a standard CAN frame. In a CAN FD long frame, the two bits following EDL, r0 and BRS, decide whether this CAN FD long frame is also a CAN FD fast frame. A CAN FD fast frame is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol.

The status bits CCCR.LACT and CCCR.FACT indicate the format of transmitted frames. When CCCR.LACT is set, frames will be transmitted in CAN FD long format with EDL = recessive. When both CCCR.LACT and CCCR.FACT are set, frames will be transmitted in CAN FD Fast format with both EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to the following table.

Table 681. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing and Prescaler Register BTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the

Fast Bit Timing and Prescaler The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter's error state. The data length code of CAN FD remote frames is transmitted as zero.

In case a M_CAN Tx Buffer is configured for CAN FD transmission with DLC > 8, the first 8 bytes are transmitted as configured in the Tx Buffer while the remaining part of the data field is padded with 0xCC. When the M_CAN receives a CAN FD frame with DLC > 8, the first 8 bytes of that frame are stored into the matching Rx Buffer or Rx FIFO. The remaining bytes are discarded.

46.3.7.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin m_can_tx the protocol controller receives the transmitted data from its local CAN transceiver via pin m_can_rx. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. [Figure 591](#) below describes how the transceiver loop delay is measured.

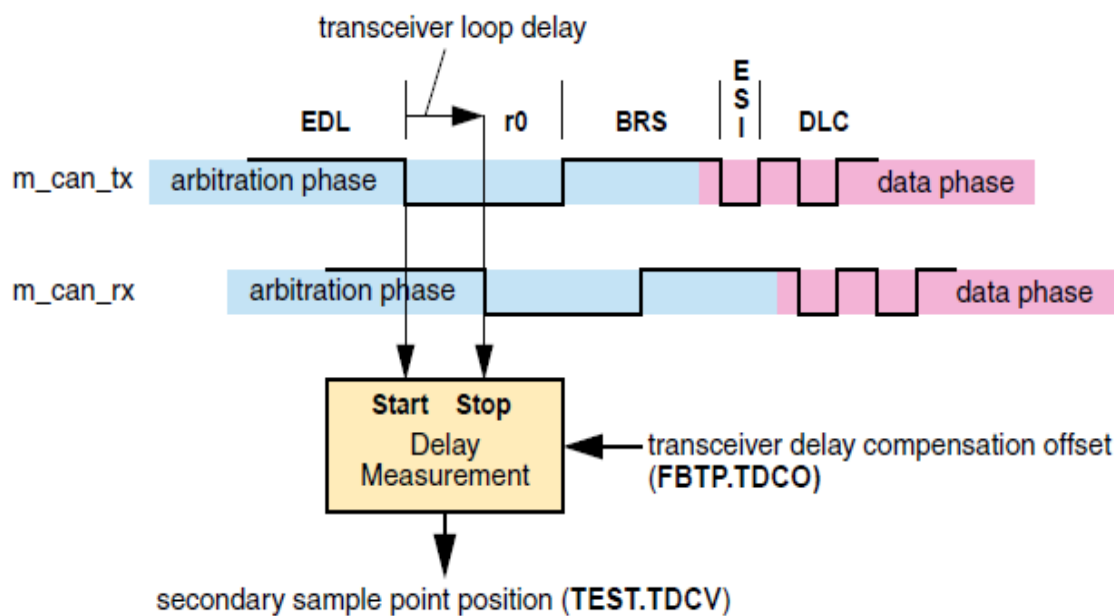


Figure 591. Transceiver delay measurement

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin m_can_tx and the data received at pin m_can_rx, starting with the falling edge of bit EDL. The delay is measured in m_can_cclk periods.

A secondary sample point is calculated by adding a configurable transceiver delay compensation offset (FBTP.TDCO) to the measured transceiver delay. The transceiver delay compensation offset is used to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta tq.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following sample point. During arbitration phase the delay compensation is always disabled.

The maximum delay which can be compensated by the M_CAN's delay compensation during the data phase is three bit times.

46.3.7.4.1 Configuration and status

Compensation for the transceiver loop delay by the M_CAN is enabled via FBTP.TDC. The transceiver delay compensation offset is configured via FBTP.TDCO. The actual delay compensation value applied by the M_CAN's protocol engine can be read from TEST.TDCV.

46.3.7.5 Bus monitoring mode

The M_CAN is set in Bus Monitoring Mode by programming CCCR[MON] to one. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode, register TXBRP is held in reset state. The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.

[Figure 592](#) shows the connection of M_CAN Tx and Rx signals to the M_CAN in Bus Monitoring Mode.

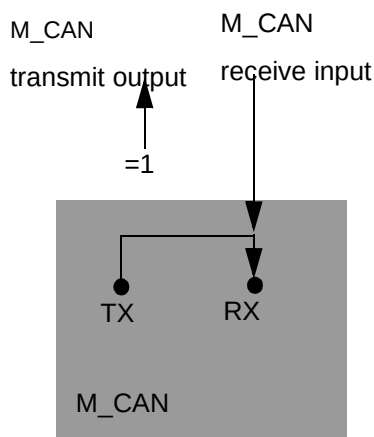


Figure 592. Pin control in Bus Monitoring mode

46.3.7.6 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

46.3.7.6.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP[TRPx] is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
 - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
 - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
 - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
 - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:
 - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
 - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

46.3.7.7 Power down (Sleep mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as one. When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module clock inputs (CAN clock and host clock) may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting signal clock stop request signal resp. CC Control Register flag CCCR[CSR]. The M_CAN will acknowledge this by resetting clock stop acknowledge output signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

46.3.7.8 Test modes

To enable write access to register TEST (see [Section 46.3.5.2.4, Test Register](#)), bit CCCR[TEST] has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin m_can_tx by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN's bit timing and it can drive constant dominant or

recessive values. The actual value at M_CAN Rx pin can be read from TEST[RX]. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST[TX] until the new configuration is visible at output Tx pin. This applies also when reading input Rx pin via TEST[RX].

Note: Test modes should be used for production tests or self test only. The software control for Tx pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

46.3.7.8.1 External Loop Back mode

The M_CAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. [Figure 593](#) shows the connection of Tx and Rx signals to the M_CAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN Rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the M_CAN Tx pin.

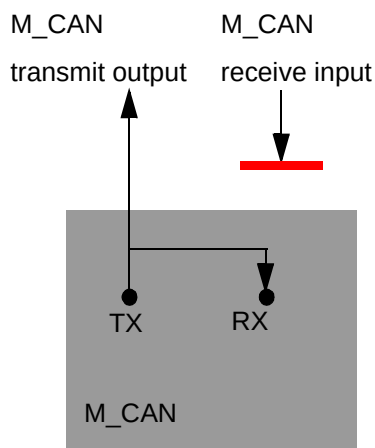


Figure 593. External Loop Back mode

46.3.7.8.2 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST[LBCK] and CCCR[MON] to one. This mode can be used for a “Hot Selftest”, meaning the M_CAN can be tested without affecting a running CAN system connected to the M_CAN Tx and Rx pins. In this mode M_CAN Tx pin is disconnected from the M_CAN and M_CAN Tx pin is held recessive. [Figure 594](#) shows the connection of M_CAN Tx pin and Rx to the M_CAN in case of Internal Loop Back mode.

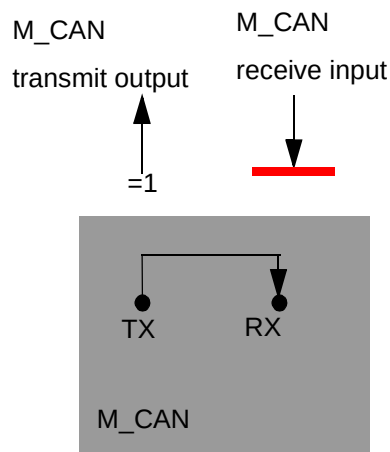


Figure 594. Internal loop back mode

46.3.8 Timestamp generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler TSCC[TCP] can be configured to clock the counter in multiples of CAN bit times (1...16). The counter is readable via TSCV[TSC]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.

46.3.9 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The Timeout Counter can only be started while CCCR[INIT] = '0'. It is stopped when CCCR[INIT] = '1', e.g. when the M_CAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: *The clock signal for the Timeout Counter is derived from the CAN core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.*

46.3.10 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

46.3.10.1 Acceptance filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
 - Range filter (from - to)
 - Filter for one or two dedicated IDs
 - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Note: When an accepted message is written to one of the two Rx FIFOs, or into a Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

46.3.10.1.1 Range filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID. There are two possibilities when range filtering is used together with extended frames:

- EFT = "00": The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = "11": The Extended ID AND Mask (XIDAM) is not used for range filtering

46.3.10.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

46.3.10.1.3 Filter for dedicated IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

46.3.10.1.4 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering. In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

46.3.10.1.5 Standard message ID filtering

[Figure 589](#) shows the flow for standard message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in [Section 46.4.6.4, Standard message ID Filter element](#).

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

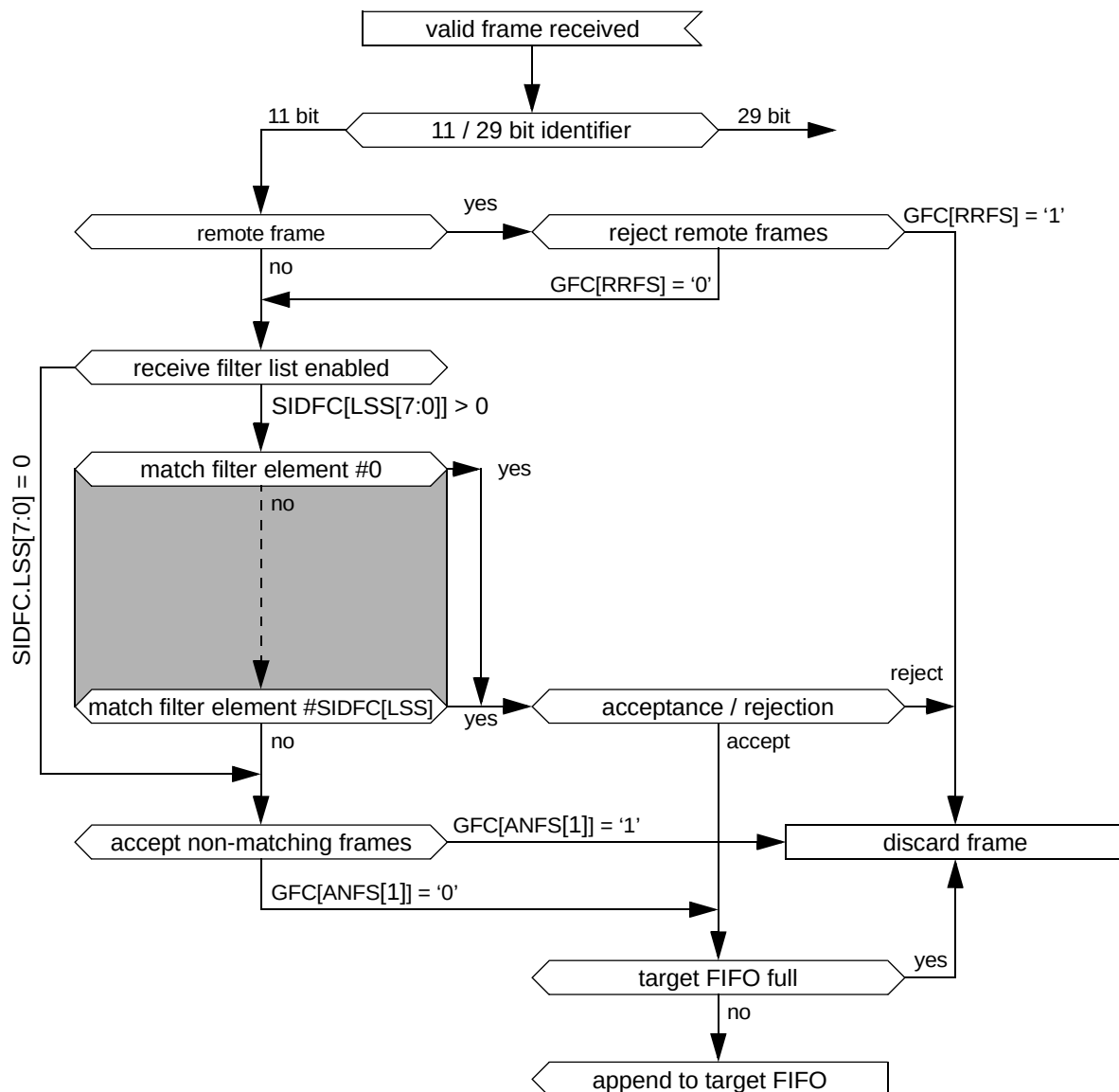


Figure 595. Standard Message ID Filter Path

46.3.10.1.6 Extended message ID filtering

[Figure 596](#) below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in [Section 46.3.5.2.20, Extended ID Filter Configuration Register \(XIDFC\)](#).

Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements which is controlled by the Global Filter Configuration (GFC) and the Extended ID Filter Configuration (XIDFC) Message ID. The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

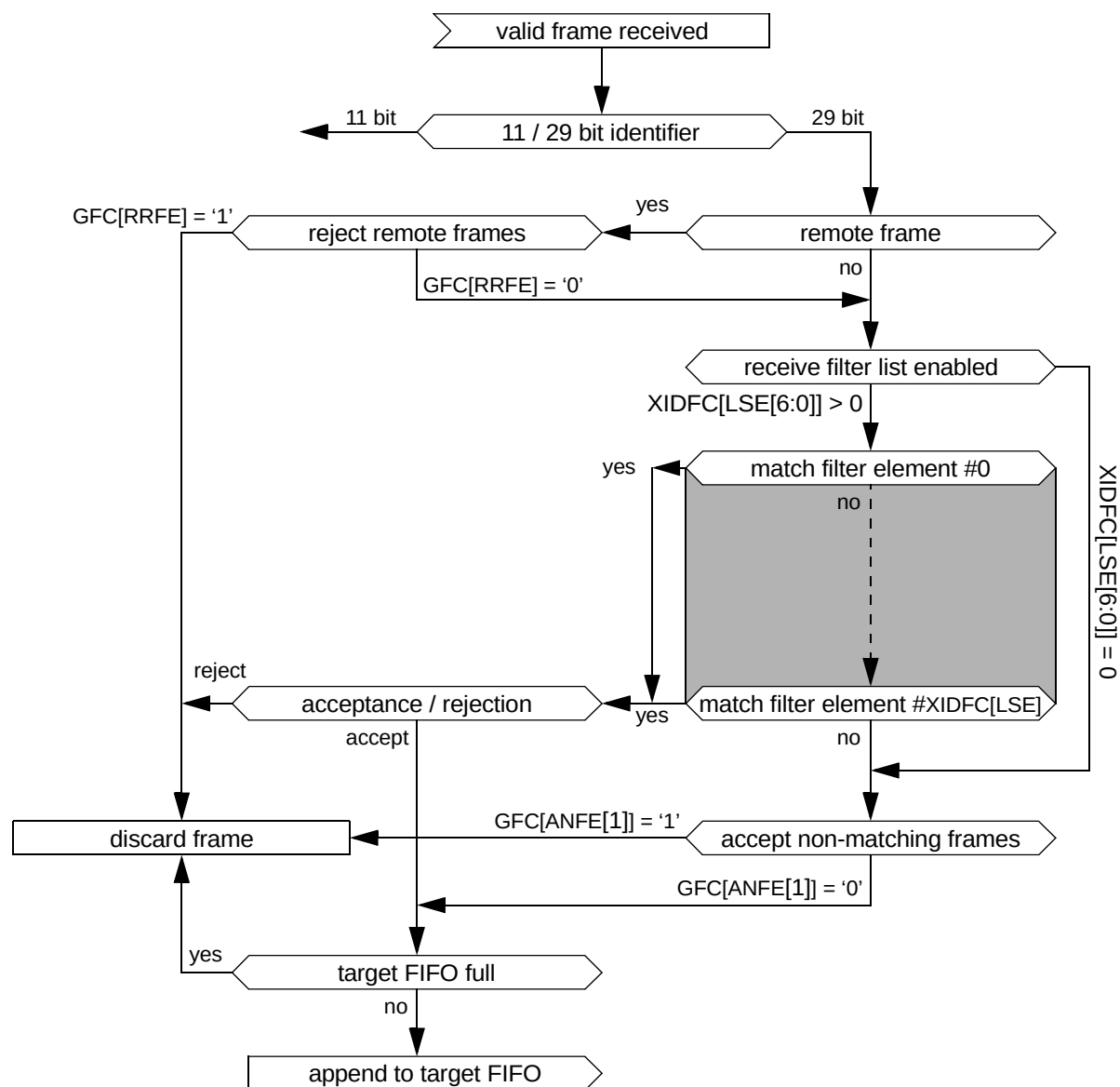


Figure 596. Extended Message ID Filter Path

46.3.10.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C. Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see [Section 46.3.10.1, Acceptance filtering](#). The Rx FIFO element is described in [Section 46.3.6.1, Rx Buffer and FIFO element](#).

When an Rx FIFO full condition is signaled by IR[RFnF], no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. In case a message is received while the corresponding

Rx FIFO is full, this message is discarded and interrupt flag IR.RFnL is set. To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC[FnWM], interrupt flag IR[RFnW] is set. When reading from an Rx FIFO, four times the Rx FIFO Get Index RXFnS[FnGI] has to be added to the corresponding Rx FIFO start address RXFnC[FnSA].

46.3.11 Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = "111" and SFID2 / EFID2[10:9] = "00" has to be configured (see [Section 46.3.6.4, "Standard message ID filter element"](#) and [Section 46.3.6.5, "Extended message ID filter element"](#)).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 682. Example Filter Configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message 1	00	00 0000
1	ID debug message 2	00	00 0001
2	ID debug message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

46.3.11.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

46.3.12 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (see [Section 46.3.6.1, "Rx Buffer and FIFO element"](#)).

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = “111” have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output m_can_dma_req is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while m_can_dma_req is activated. The behavior is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets m_can_dma_ack. This resets m_can_dma_req. Now the M_CAN is prepared to receive the next set of debug messages.

46.3.12.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to “111”. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

Table 683. Example Filter Configuration for debug messages

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

46.3.12.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S.DMS.

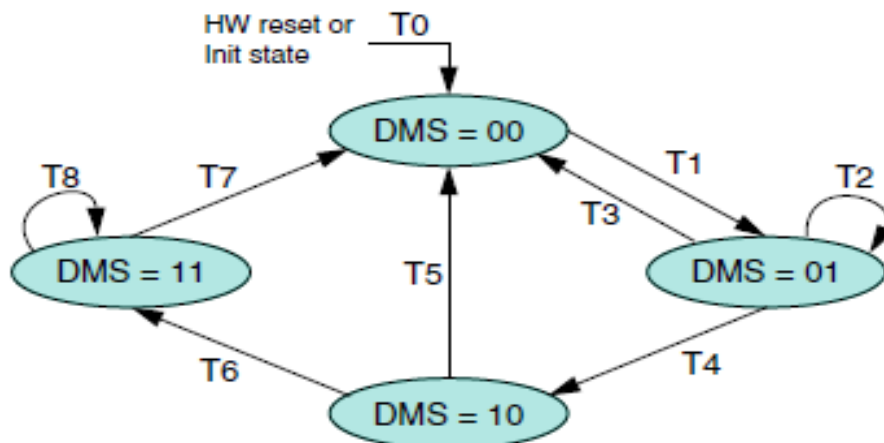


Figure 597. Debug Message Handling State Machine

- T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C
- T1: reception of debug message A
- T2: reception of debug message A
- T3: reception of debug message C
- T4: reception of debug message B
- T5: reception of debug messages A, B
- T6: reception of debug message C
- T7: DMA transfer completed
- T8: reception of debug message A,B,C (message rejected)

46.3.13 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in [Section 46.3.6.2, Tx buffer element](#).

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

46.3.13.1 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional

Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of a Dedicated Tx Buffer in the Message RAM is calculated by adding four times the transmit buffer index (0...31) to the Tx Buffer Start Address TXBC[TBSA].

46.3.13.2 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF] = '1') is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFOs Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level. When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding four times the Put Index TXFQS[TFQPI] (0...31) to the Tx Buffer Start Address TXBC[TBSA].

46.3.13.3 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first. New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF] = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

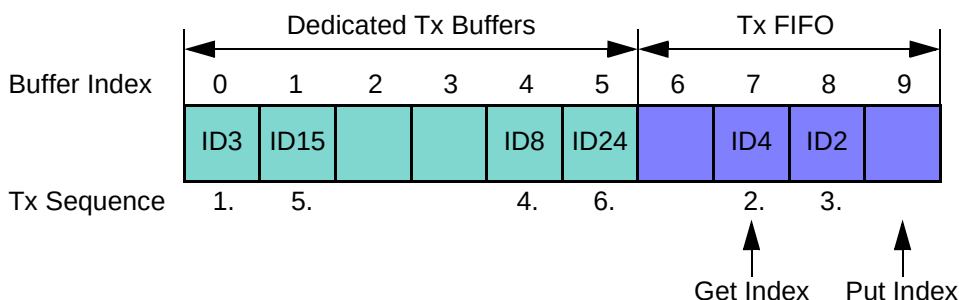
The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding four times the Tx Queue Put Index $\text{TXFQS}[\text{TFQPI}]$ (0...31) to the Tx Buffer Start Address $\text{TXBC}[\text{TBSA}]$.

46.3.13.4 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by $\text{TXBC}[\text{NDTB}]$. The number of Tx Buffers assigned to the Tx FIFO is configured by $\text{TXBC}[\text{TFQS}]$. In case $\text{TXBC}[\text{TFQS}]$ is programmed to zero, only Dedicated Tx Buffers are used.

Figure 598. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO



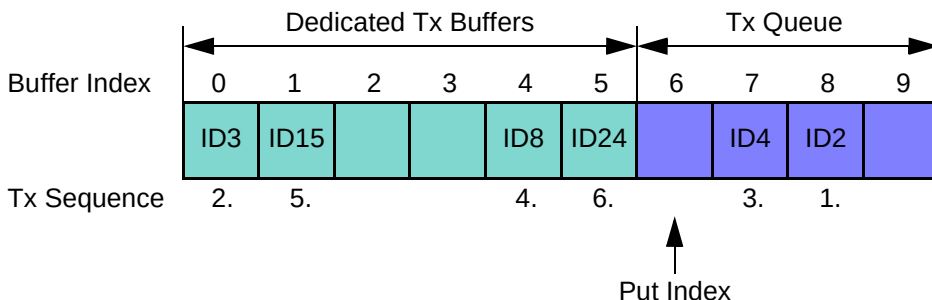
Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by $\text{TXFS}[\text{TFGI}]$)
- Buffer with lowest Message ID gets highest priority and is transmitted next

46.3.13.5 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by $\text{TXBC}[\text{NDTB}]$. The number of Tx Queue Buffers is configured by $\text{TXBC}[\text{TFQS}]$. In case $\text{TXBC}[\text{TFQS}]$ is programmed to zero, only Dedicated Tx Buffers are used.

Figure 599. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue



Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

46.3.13.6 Transmit cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

46.3.13.7 Tx Event handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in [Section 46.3.6.3, Tx event FIFO element](#). When a Tx Event FIFO full condition is signaled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

46.3.14 FIFO acknowledge handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see [Section 46.3.5.2.27, Rx FIFO 0 Acknowledge Register \(RXF0A\)](#), [Section 46.3.5.2.31, Rx FIFO 1 Acknowledge Register \(RXF1A\)](#), and [Section 46.3.5.2.43, Tx Event FIFO Acknowledge Register \(TXEFA\)](#)). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFOs Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFOs Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFOs Fill Level. In this case some of the older FIFO elements would be lost.

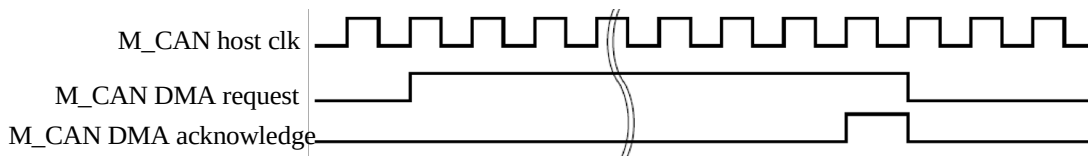
Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

46.3.15 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_CAN DMA request signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_CAN while M_CAN DMA request signal is active.

After the transfer of the received messages has completed the DMA unit activates the M_CAN DMA ACK signal. This resets M_CAN DMA request signal. The debug message handling state machine enters idle state (DMS = "00") and waits for reception of the next debug messages.

Figure 600. Timing of DMA interface signals



Note: If the DMA unit activates input M_CAN DMA acknowledge signal before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

46.4 Time Triggered Modular CAN (M_TTCAN) core

The M_TTCAN module is the new M_TTCAN Communication Controller IP-module that can be integrated as stand-alone device or as part of an ASIC. The M_TTCAN provides all features of time-triggered communication specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation.

In addition the M_TTCAN supports communication according to CAN FD protocol specification 1.0 with data fields of up to eight bytes length. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_TTCAN via the Generic Master Interface. All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information. It implements all functions concerning the time schedule and the global system time.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_TTCAN's clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

46.4.1 Features

The following are the features of M_TTCAN:

- M_TTCAN protocol level 1 and level 2 completely in hardware
- AUTOSAR optimized
- SAE J1939 optimized
- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1, -4
- CAN FD with max. 8 data bytes supported
- Event synchronized time-triggered communication supported
- CAN Error Logging
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 Transmit Buffers
- Configurable Transmit FIFO / Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_(TT)CAN share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 32 bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support

46.4.2 Block Diagram

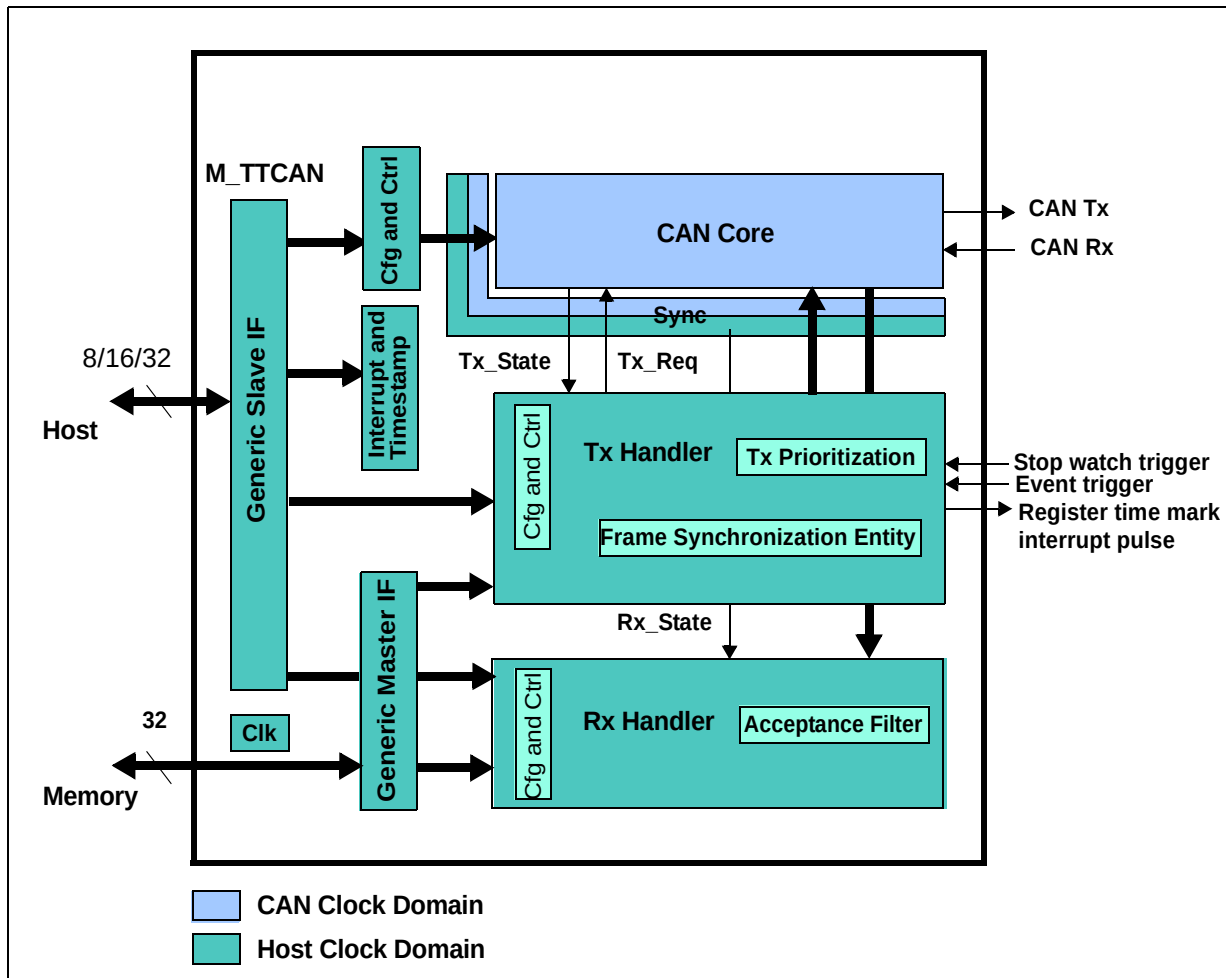


Figure 601. M_TTCAN core block diagram

- CAN core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Cfg and Ctrl: CAN core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported. The Tx Handler also implements the Frame Synchronization Entity FSE which controls time-triggered communication according to ISO11898-4. It synchronizes itself to the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the

predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the Message RAM. Stop Watch Trigger, Event Trigger, and Time Mark Interrupt are synchronization interfaces.

- **Rx Handler:** Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs
- **Clk:** Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- **Generic Slave Interface:** Connects the M_TTCAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- **Generic Master Interface:** Connects the M_TTCAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB × 32 bit.

46.4.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as M_TTCAN clock. The CAN core should be programmed to have at least 8 clocks per bit time, this is e.g. 1 Mbaud at M_TTCAN clock ≥ 8 MHz. Even in case of a very high Host clock frequency, the clock frequency of the CAN core needs not to be higher than 8 MHz.

Within the M_TTCAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note: *In order to achieve a stable function of the M_TTCAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.*

Caution: It is strongly recommended to stop this module when the device is in the STOP mode or in case, if the CAN message reception is mandatory, please check if the PLL is unlocked before transmitting any message, or use the external oscillator instead of the the IRC during the STOP mode.

46.4.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to M_TTCAN INT 0 or to M_TTCAN INT 1. By default all interrupts are routed to interrupt line M_TTCAN INT 0. By programming INT 0 and INT 1 bits of the Interrupt Line Enable (ILE) register, the interrupt lines can be enabled or disabled separately.

46.4.5 Memory map and registers description

46.4.5.1 Hardware reset description

After hardware reset, the registers of the M_TTCAN hold the reset values listed in [Table 684](#). Additionally the Bus_Off state is reset and the CAN transmit output is set to recessive (HIGH). The value 0x0001 (CCCR[INIT] = '1') in the CC Control Register enables

software initialization. The M_TTCAN does not influence the CAN bus until the CPU resets CCCR[INIT] to '0'.

46.4.5.2 Register map

The M_TTCAN module allocates an address space of 512 bytes. All registers are organized as 32-bit registers. The M_TTCAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

Note: Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only with CCCR[CCE]= '1' AND CCCR[INIT]= '1'. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

Table 684. M_TTCAN Memory Map

Address Offset	Register Name	Access	Reset value ⁽¹⁾	Section/Page
0x0000	CREL—Core Release Register	R	rrrd dddd	46.4.5.2.1/1462
0x0004	ENDN—Endian Register	R	8765 4321	46.4.5.2.2/1463
0x0008	Reserved	—	—	—
0x000C	FBTP—Fast Bit Timing and Prescaler Register	RP	0x000_0A33	46.4.5.2.3/1463
0x0010	TEST—Test Register	RW	0x0000_0000	46.4.5.2.4/1465
0x0014	RWD—RAM Watchdog Register	RP	0x0000_0000	46.4.5.2.5/1466
0x0018	CCCR—CC Control Register	RP	0x0000_0001	46.4.5.2.6/1467
0x001C	BTP—Bit Timing and Prescaler Register	RP	0x0000_0A33	46.4.5.2.7/1469
0x0020	TSCC—Timestamp Counter Configuration Register	RP	0x0000_0000	46.4.5.2.8/1471
0x0024	TSCV—Timestamp Counter Value Register	W1C	0x0000_0000	46.4.5.2.9/1472
0x0028	TOCC—Timeout Counter Configuration Register	RP	0x0000_0000	46.4.5.2.10/1473
0x002C	TOCV—Timeout Counter Value Register	W1C	0x0000_FFFF	46.4.5.2.11/1474
0x0030–0x003C	Reserved	—	—	—
0x0040	ECR—Error Counter Register	R	0x0000_0000	46.4.5.2.12/1474
0x0044	PSR—Protocol Status Register	RS	0x0000_0707	46.4.5.2.13/1475
0x0048–0x004C	Reserved	—	—	—
0x0050	IR—Interrupt Register	RW	0x0000_0000	46.4.5.2.14/1477
0x0054	IE—Interrupt Enable Register	RW	0x0000_0000	46.4.5.2.15/1481
0x0058	ILS—Interrupt Line Select Register	RW	0x0000_0000	46.4.5.2.16/1484
0x005C	ILE—Interrupt Line Enable Register	RW	0x0000_0000	46.4.5.2.17/1487
0x0060–0x007C	Reserved	—	—	—
0x0080	GFC—Global Filter Configuration Register	RP	0x0000_0000	46.4.5.2.18/1488

Table 684. M_TTCAN Memory Map (continued)

Address Offset	Register Name	Access	Reset value ⁽¹⁾	Section/Page
0x0084	SIDFC—Standard ID Filter Configuration Register	RP	0x0000_0000	46.4.5.2.19/1489
0x0088	XIDFC—Extended ID Filter Configuration Register	RP	0x0000_0000	46.4.5.2.20/1490
0x008C	Reserved	—		
0x0090	XIDAM—Extended ID AND Mask Register	RP	0x1FFF_FFFF	46.4.5.2.21/1491
0x0094	HPMS—High Priority Message Status Register	R	0x0000_0000	46.4.5.2.22/1492
0x0098	NDAT1—New Data 1 Register	RW	0x0000_0000	46.4.5.2.23/1493
0x009C	NDAT2—New Data 2 Register	RW	0x0000_0000	46.4.5.2.24/1493
0x00A0	RXF0C—Rx FIFO 0 Configuration Register	RP	0x0000_0000	46.4.5.2.25/1494
0x00A4	RXF0S—Rx FIFO 0 Status Register	R	0x0000_0000	46.4.5.2.26/1495
0x00A8	RXF0A—Rx FIFO 0 Acknowledge Register	RW	0x0000_0000	46.4.5.2.27/1496
0x00AC	RXBC—Rx Buffer Configuration Register	RW	0x0000_0000	46.4.5.2.28/1496
0x00B0	RXF1C—Rx FIFO 1 Configuration Register	RP	0x0000_0000	46.4.5.2.29/1497
0x00B4	RXF1S—Rx FIFO 1 Status Register	R	0x0000_0000	46.4.5.2.30/1498
0x00B8	RXF1A—Rx FIFO 1 Acknowledge Register	RW	0x0000_0000	46.4.5.2.31/1499
0x00BC	Reserved	—		
0x00C0	TXBC—Tx Buffer Configuration Register	RP	0x0000_0000	46.4.5.2.32/1499
0x00C4	TXFQS—Tx FIFO/Queue Status Register	R	0x0000_0000	46.4.5.2.33/1500
0x00C8	Reserved	—		
0x00CC	TXBRP—Tx Buffer Request Pending Register	R	0x0000_0000	46.4.5.2.34/1501
0x00D0	TXBAR—Tx Buffer Add Request Register	RW	0x0000_0000	46.4.5.2.35/1502
0x00D4	TXBCR—Tx Buffer Cancellation Register	RW	0x0000_0000	46.4.5.2.36/1503
0x00D8	TXBTO—Tx Buffer Transmission Occurred Register	R	0x0000_0000	46.4.5.2.37/1504
0x00DC	TXBCF—Tx Buffer Cancellation Finished Register	R	0x0000_0000	46.4.5.2.38/1504
0x00E0	TXBTIE—Tx Buffer Transmission Interrupt Enable	RW	0x0000_0000	46.4.5.2.39/1505
0x00E4	TXBCIE—Tx Buffer Cancellation Finished Interrupt Enable Register	RW	0x0000_0000	46.4.5.2.40/1506
0x00E8–0x00EC	Reserved	—		
0x00F0	TXEFC—Tx Event FIFO Configuration Register	RP	0x0000_0000	46.4.5.2.41/1506
0x00F4	TXEFS—Tx Event FIFO Status Register	R	0x0000_0000	46.4.5.2.42/1507
0x00F8	TXEFA—Tx Event FIFO Acknowledge Register	RW	0x0000_0000	46.4.5.2.43/1508

Table 684. M_TTCAN Memory Map (continued)

Address Offset	Register Name	Access	Reset value ⁽¹⁾	Section/Page
0x00FC	Reserved	—		
0x100	TTTMC—TT Trigger Memory Configuration Register	RP	0x0000_0000	46.4.5.2.44/1509
0x104	TTRMC—TT Reference Message Configuration Register	RP	0x0000_0000	46.4.5.2.45/1510
0x108	TTOCF—TT Operation Configuration Register	RP	0x0001_0000	46.4.5.2.46/1511
0x10C	TTMLM—TT Matrix Limits Register	RP	0x0000_0000	46.4.5.2.47/1512
0x110	TURCF—TUR Configuration Register	RP	0x1000_0000	46.4.5.2.48/1513
0x114	TTOCN—TT Operation Control Register	RW	0x0000_0000	46.4.5.2.49/1515
0x118	TTGTP—TT Global Time Preset Register	RW	0x0000_0000	46.4.5.2.50/1517
0x11C	TTTMK—TT Time Mark Register	RW	0x0000_0000	46.4.5.2.51/1518
0x120	TTIR—TT Interrupt Register	RW	0x0000_0000	46.4.5.2.52/1519
0x124	TTIE—TT Interrupt Enable Register	RW	0x0000_0000	46.4.5.2.53/1521
0x128	TTILS—TT Interrupt Line Select Register	RW	0x0000_0000	46.4.5.2.54/1523
0x12C	TTOST—TT Operation Status Register	R	0x0000_0080	46.4.5.2.55/1525
0x130	TURNA—TUR Numerator Actual Register	R	0x0001_0000	46.4.5.2.56/1527
0x134	TTLGT—TT Local and Global Time Register	R	0x0000_0000	46.4.5.2.57/1528
0x138	TTCTC—TT Cycle Time and Count Register	R	0x003F_0000	46.4.5.2.58/1528
0x13C	TTCPT—TT Capture Time Register	R	0x0000_0000	46.4.5.2.59/1529
0x140	TTCSM—TT Cycle Sync Mark Register	R	0x0000_0000	46.4.5.2.60/1530
0x144–1FC	Reserved	—		

1. R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected Write, C = Clear/preset on write, r = release, d = date, u =undefined

46.4.5.2.1 Core Release Register (CREL)

Figure 602. Core Release register

Address: 0x0000

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset ⁽¹⁾	r	r	r	r	r	r	r	r	r	r	r	r	d	d	d	d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset ¹	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d

1. R = Read; -r = release, -d = time stamp. The coding of revisions depends on the module version used in the device.

Table 685. Core Release register field descriptions

Field	Description
0:3 REL[3:0]	Core Release One digit, BCD.
4:7 STEP[3:0]	Step of Core Release One digit, BCD.
8:11 SUBSTEP[3:0]	Sub-step of Core Release One digit, BCD.
12:15 YEAR[3:0]	Time Stamp Year One digit, BCD..
16:23 MON[7:0]	Time Stamp Month Two digits, BCD.
24:31 DAY[7:0]	Time Stamp Day Two digits, BCD.

46.4.5.2.2 Endian Register (ENDN)

Figure 603. Endian Register

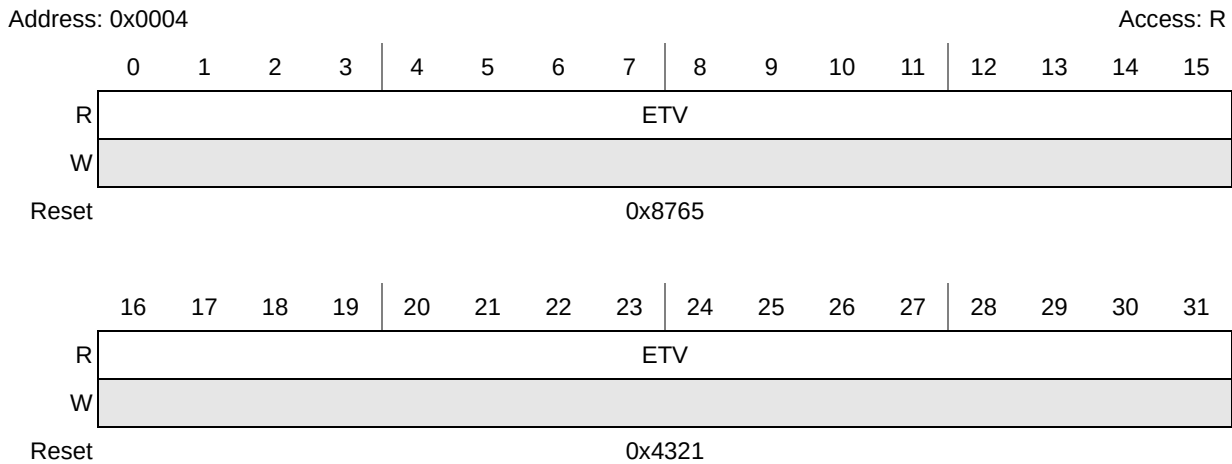


Table 686. Endian Register Field Descriptions

Field	Description
0:31	Endianness Test Value
ETV[31:0]	The endianness test value is 0x87654321.

46.4.5.2.3 Fast Bit Timing and Prescaler Register (FBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 M_TTCAN clock periods. $tq = (FBRP + 1) M_TTCAN$ clock period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. **FTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (prog rammed values) **[FTSEG1 + FTSEG2 + 3]** tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Figure 604. Fast Bit Timing and Prescaler Register

Address: 0x000C

Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TDCO ⁽¹⁾				TDC ⁽¹⁾	0	FBRP ⁽¹⁾									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	FTSEG1 ⁽¹⁾				0	FTSEG2 ⁽¹⁾				0	FSJW ⁽¹⁾	
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 687. Fast Bit Timing and Prescaler Register

Field	Description
0:3 TDCO	Transceiver Delay Compensation Offset (0x0–0xF)— Offset value defining the distance between the measured delay from M_TTCAN transmit to M_TTCAN receive and the secondary sample point. Valid values are 0 to 15 M_TTCAN clock periods.
4 TDC	Transceiver Delay Compensation 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
6:15 FBRP	Fast Baud Rate Prescaler (0x000–0x3FF)— The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
20:23 FTSEG1	Fast time segment before sample point (0x1–0xF) Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
25:27 FTSEG2	Fast time segment before sample point (0x0–0x7) Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
30:31 FSJW	Fast (Re) Synchronization Jump Width (0x0–0x3) Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a M_TTCAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_TTCAN for a fast bit rate of 500 kbit/s.

46.4.5.2.4 Test Register (TEST)

Write access to the Test Register has to be enabled by setting bit CCCR[TEST] to '1'. All Test Register functions are set to their reset values when bit CCCR[TEST] is reset.

Loop Back mode and software control of Tx pin are hardware test modes. Programming of TX ≠ "00" may disturb the message transfer on the CAN bus.

Figure 605. Test Register

Address: 0x0010																Access: RP			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0			TDCV					RX	TX ⁽¹⁾			LBCK ⁽¹⁾	CAT	CAM	TAT ⁽¹⁾	TAM ⁽¹⁾
W																	
Reset	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 688. Test Register field descriptions

Field	Description
0:18	Reserved
19:23 TDCV	Transceiver Delay Compensation Value (0x00–0x1F) Position of the secondary sample point, defined by the sum of the measured delay from M_TTCAN transmit to receive and FBTP.TDCO. Valid values are 0 to 31 M_TTCAN core clock periods.
24 RX	Receive Pin Monitors the actual value of transmit pin 0 The CAN bus is dominant 1 The CAN bus is recessive
25:26 TX	Control of Transmit Pin 00 Reset value TX is controlled by the CAN core, updated at the end of the CAN bit time 01 Sample Point can be monitored at pin 10 Dominant ('0') level at pin 11 Recessive ('1') at pin
27 LBCX	Loop Back mode 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (see Section 46.4.7.9, Test modes)
28 CAT	Check ASC Transmit Control Monitors level at M_TTCAN ASC transmit control. 0 Output M_TTCAN ASC transmit control pin = '0' 1 Output M_TTCAN ASC transmit control pin = '1'

Table 688. Test Register field descriptions

Field	Description
29 CAM	Check ASC Multiplexer Control Monitors level at M_TTCAN ASC multiplexer control. 0 Output M_TTCAN ASC multiplexer control pin = '0' 1 Output M_TTCAN ASC multiplexer control pin = '1'
30 TAT	Test ASC Transmit Control Controls output ASC Transmit Control pin in test mode, ORed with the signal from the FSE 0 Level at M_TTCAN ASC transmit control pin controlled by FSE 1 Level at M_TTCAN ASC transmit control pin = '1'
31 TAM	Test ASC Multiplexer Control Controls output M_TTCAN ASC multiplexer control in test mode, ORed with the signal from the FSE 0 Level at M_TTCAN ASC multiplexer control pin controlled by FSE 1 Level at M_TTCAN ASC multiplexer control pin = '1'

46.4.5.2.5 RAM Watchdog Register (RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the **RWD[WDC]** bits. The counter is reloaded with **RWD[WDC]** bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag **IR[WDI]** bit is set. The RAM Watchdog Counter is clocked by the Host clock.

Figure 606. RAM watchdog register

Address: 0x0014																Access: RP	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0																
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDV								WDC ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bit is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 689. RAM Watchdog register field descriptions

Field	Description
0:15	Reserved
16:23 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value.
24:31 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled.

46.4.5.2.6 CC Control Register (CCCR)

For details about setting and resetting of single bits see [Section 46.4.7.1.1, Software initialization](#).

Figure 607. CC Control register

Address: 0x0018

Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FACT	LACT	CMR		CME ⁽¹⁾		TEST ⁽¹⁾	DAR ⁽¹⁾	MON ⁽¹⁾	CSR	CSA	ASM ⁽¹⁾	CCE ⁽¹⁾	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1"

Table 690. CC Control Register Field Descriptions

Field	Description
0:17	Reserved
18 FACT	Fast Frame Mode Active 0 Format of transmitted frames defined by LACT 1 Frames transmitted in CAN FD Fast format
19 LACT	Long Frame Mode Active 0 Frames transmitted in Standard CAN format 1 Frames transmitted in CAN FD Long format

Table 690. CC Control Register Field Descriptions

Field	Description
20:21 CMR	CAN Mode Request A change of the CAN operation mode is requested by writing to this bit field. After change to the requested operation mode the bit field is reset to "00" and the status flags FACT and LACT are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CMR is retained until it is overwritten by the next mode change request. Default is normal CAN operation. 00 unchanged 01 Long Frame Mode request 10 Long + Fast Frame Mode request 11 Normal CAN operation request
22:23 CME	CAN Mode Enable 00 Normal CAN operation according to ISO11898-1 01 Long Frame Mode enabled 10 Long + Fast Frame Mode enabled 11 Long + Fast Frame Mode enabled <i>Note: In case CME = "01" transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CME = "10"/"11" transmission and reception of long and fast CAN FD frames is enabled.</i>
24 TEST	Test Mode Enable 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	DAR: Disable Automatic Retransmission 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_TTCAN may be set in power down by stopping host clock and core clock

Table 690. CC Control Register Field Descriptions

Field	Description
29 ASM	ASM Restricted Operation Mode The Restricted Operation Mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame. In the optional Restricted Operation Mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. If the M_CAN is connected to a Clock Calibration on CAN unit, ASM is controlled by input M_CAN calibration OK (m_can_cok) signal. In case "m_can_cok" switches to '0', bit ASM is set. When "m_can_cok" switches back to '1', bit ASM returns to the previously written value. The state of ASM is the written value while input "m_can_cok" is at '1'. The input is hardwired to '1' when there is no Clock Calibration on CAN unit connected. 0 Normal CAN operation 1 Restricted Operation Mode active
30 CCE	Configuration Change Enable 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = '1')
31 INIT	Initialization 0 Normal Operation 1 Initialization is started

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

46.4.5.2.7 Bit Timing and Prescaler Register (BTP)

This register is only writable if bits **CCCR[CCE]** and **CCCR[INIT]** are set. The CAN bit time may be programmed in the range of [4...25] time quanta (tq). The CAN time quantum may be programmed in the range of [1...1024] M_TTCAN core clockperiods.

$$\text{Equation 28} \quad tq = (BRP + 1) M_TTCAN \text{ clock period}$$

where $BRP = (0x40 \cdot BRPE) + BRPL$.

TSEG1 is the sum of Prop_Seg and Phase_Seg1. **TSEG2** is Phase_Seg2. Therefore the length of the bit time is (programmed values) $[TSEG1 + TSEG2 + 3] tq$ or (functional values) $[Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq$.

Figure 608. Bit Timing and Prescaler Register

Address: 0x001C

Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	BRP ⁽¹⁾									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	TSEG1 ⁽¹⁾						TSEG2 ⁽¹⁾				SJW ⁽¹⁾			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 691. Bit Timing and Prescaler Field Descriptions

Field	Description
0:5	Reserved
6:15 BRPE	Baud Rate Prescaler Extension (0x000–0x3FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16:17	Reserved
18:23 TSEG1	(0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24:27 TSEG2	(0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28:31 SJW	(Re) Synchronization Jump Width (0x0–0x3) valid programmed values are 0-3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used

Note: With a CAN clock of 8 MHz, the reset value of 0x00002301 configures the M_TTCAN for a bit rate of 500 kbit/s.

Note: The Information Processing Time (IPT) of the M_TTCAN is 0 tq. Therefore, to conform to the CAN protocol specification, the lengths for phase segment 1 (Phase_Seg1) and phase segment 2 (Phase_Seg2) must be equal.

46.4.5.2.8 Timestamp Counter Configuration Register (TSCC)

Figure 609. Timestamp Counter Configuration Register

Address: 0x0020

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												TCP ⁽¹⁾			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												TSS ⁽¹⁾			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bit is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 692. Timestamp Counter Configuration Field Descriptions

Field	Description
0:11	Reserved
12:15 TCP	Timestamp Counter Prescaler 0x0–0xF Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1...16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. <i>Note: With CAN FD an external counter is required for timestamp generation (TSS = "10")</i>
16:29	Reserved
30:31 TSS	Timestamp Select 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to TCP 10 Reserved 11 Same as "00"

46.4.5.2.9 Timestamp Counter Value Register (TSCV)

Figure 610. Timestamp counter value register

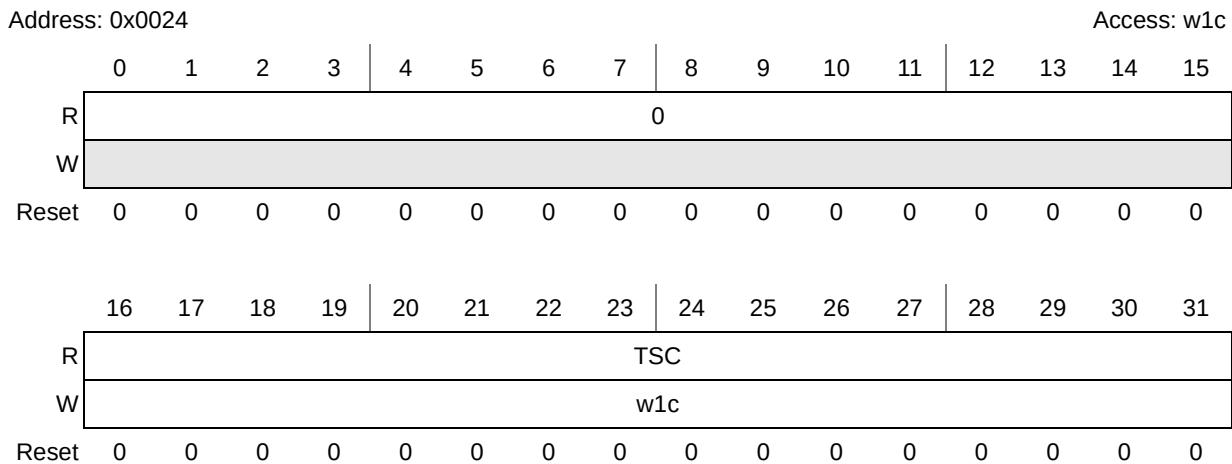


Table 693. Timestamp counter value field descriptions

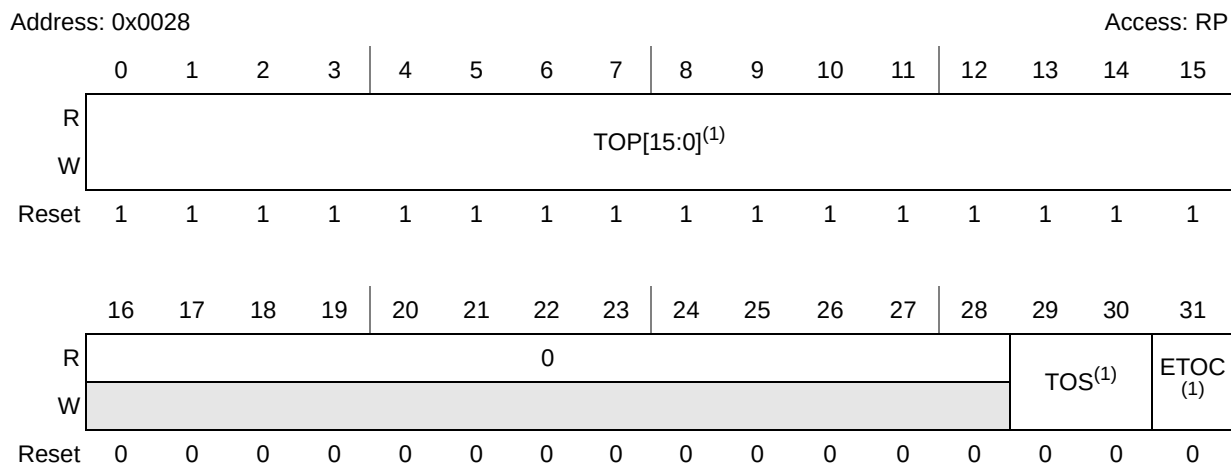
Field	Description
0:15	Reserved
16:30 TSC	Timestamp Counter The Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC[TSS] = "01", the Timestamp Counter is incremented in multiples of CAN bit times [1...16] depending on the configuration of TSCC[TCP] . A wrap around sets interrupt flag IR[TSW] . Write access resets the counter to zero.

Note: A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

46.4.5.2.10 Timeout Counter Configuration Register (TOCC)

For a description of the Timeout Counter see [Section 46.4.7.11, Timeout counter](#).

Figure 611. Timeout counter configuration register



1. These are protected write (P) bits which means that write access by the bit is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 694. Timeout counter configuration field descriptions

Field	Description
0:15 TOP[15:0]	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
16:28	Reserved
29:30 TOS	Timeout Select When operating in Continuous mode, a write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP] . Down-counting is started when the first FIFO element is stored. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counter 0 Timeout Counter disabled 1 Timeout Counter enabled <i>Note: For use of timeout function with CAN FD see Section 46.4.7.11, "Timeout counter".</i>

46.4.5.2.11 Timeout Counter Value Register (TOCV)

Figure 612. Timeout counter value register

Address: 0x002C

Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TOC															
W	w1c															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 695. Timeout counter value field descriptions

Field	Description
0:15	Reserved
16:30 TOC	Timeout Counter The Timeout Counter is decremented in multiples of CAN bit times [1...16] depending on the configuration of TSCC.TCP . When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS .

46.4.5.2.12 Error Counter Register (ECR)

Figure 613. Error counter register

Address: 0x0040

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								CEL ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Access type is RX: reset on read.

Table 696. Error counter register field descriptions

Field	Description
0:7	Reserved
8:15 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL . The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR[ELO] .
16 RP	Receive Error Passive 0 The Receive Error Counter is below the <i>error passive</i> level of 128 1 The Receive Error Counter has reached the <i>error passive</i> level of 128
17:23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127
25:31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255 <i>Note: When CCCR[ASM] is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented. This enables monitoring of collisions between CAN frames and ASC frames.</i>

46.4.5.2.13 Protocol Status Register (PSR)

Figure 614. Protocol status register (PSR)

Address: 0x0044

Access: RS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	REDL (1)	RBRSL (1)	RESI (1)	FLEC ⁽²⁾			BO	EW	EP	ACT		LEC ⁽²⁾		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

1. Access type is RX: reset on read.

2. Access type is RS: set on read.

Table 697. PSR descriptions

Field	Description
0:17	Reserved

Table 697. PSR descriptions (continued)

Field	Description
18 REDL	Received CAN FD Message 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with EDL flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not have its BRS flag set. 1 Last received CAN FD message had its BRS flag set.
20 RESI	ESI CAN FD Message with ESI flag This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not have its ESI flag set. 1 Last received CAN FD message had its ESI flag set
21 FLEC	Fast Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.
24 BO	Bus_Off Status 0 The M_TTCAN is not Bus_Off 1 The M_TTCAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_TTCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_TTCAN is in the Error_Passive state
27:28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing — node is synchronizing on CAN communication 01 Idle — node is neither receiver nor transmitter 10 Receiver — node is operating as receiver 11 Transmitter — node is operating as transmitter

Table 697. PSR descriptions (continued)

Field	Description
29:31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_TTCAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCErrror: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT]. If the device goes Bus_Off, it will set CCCR.INIT of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129×11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

46.4.5.2.14 Interrupt Register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of **IE** controls whether an interrupt is generated. The configuration of **ILS** controls on which interrupt line an interrupt is signaled.

Figure 615. Interrupt register

Address: 0x0050

Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STE	FOE	ACKE	BE	CRCE	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	UMD	TSW
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 698. Interrupt register field descriptions

Field	Description
0 STE	Stuff Error 0 No Stuff Error detected 1 More than 5 equal bits in a sequence occurred
1 FOE	Format Error 0 No Format Error detected 1 A fixed format part of a received frame has the wrong format
2 ACKE	Acknowledge Error 0 No Acknowledge Error detected 1 A transmitted message was not acknowledged by another node
3 BE	Bit Error 0 No Bit Error detected 1 Device wanted to send a <i>rec</i> / <i>dom</i> level, but monitored bus level was <i>dom</i> / <i>rec</i>
4 CRCE	CRC Error 0 No CRC Error detected 1 Received CRC did not match the calculated CRC
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed

Table 698. Interrupt register field descriptions (continued)

Field	Description
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to '1'. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (e.g. parity logic)
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (e.g. ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 UMD	Unprocessed Message Discarded When a new message is received while the acceptance filtering process for the previously received message has not yet completed, this message is discarded. 0 No unprocessed message discarded 1 Unprocessed message discarded
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Element Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark
19 TEFN	Tx Event FIFO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element

Table 698. Interrupt register field descriptions (continued)

Field	Description
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

46.4.5.2.15 Interrupt Enable Register (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signaled on an interrupt line.

Figure 616. Interrupt enable register

Address: 0x0054

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	UMDE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLE	TEFFE	TEFWE	TEFNE	TTEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 699. Interrupt enable register field descriptions

Field	Description
0 STEE	Stuff Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
1 FOEE	Format Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
2 ACKEE	Acknowledge Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
3 BEE	Bit Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
4 CRCEE	CRC Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 699. Interrupt enable register field descriptions (continued)

Field	Description
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 UMDE	Unprocessed Message Discarded Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 699. Interrupt enable register field descriptions (continued)

Field	Description
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

46.4.5.2.16 Interrupt Line Select Register (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1].

Figure 617. Interrupt line select register

Address: 0x0058

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	UMDL	TSWL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RFILL	RFIFL	RFIWL	RFINL	RFOLL	RF0FL	RF0WL	RF0NL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 700. Interrupt line select register field descriptions

Field	Description
0 STEL	Stuff Error Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
1 FOEL	Format Error Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
2 ACKEL	Acknowledge Error Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
3 BEL	Bit Error Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
4 CRCEL	CRC Error Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1

Table 700. Interrupt line select register field descriptions (continued)

Field	Description
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
12 UMDL	UMDL: Unprocessed Message Discarded Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
16 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1

Table 700. Interrupt line select register field descriptions (continued)

Field	Description
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1

46.4.5.2.17 Interrupt Line Enable Register (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits **EINT0** and **EINT1**.

Figure 618. Interrupt line enable register

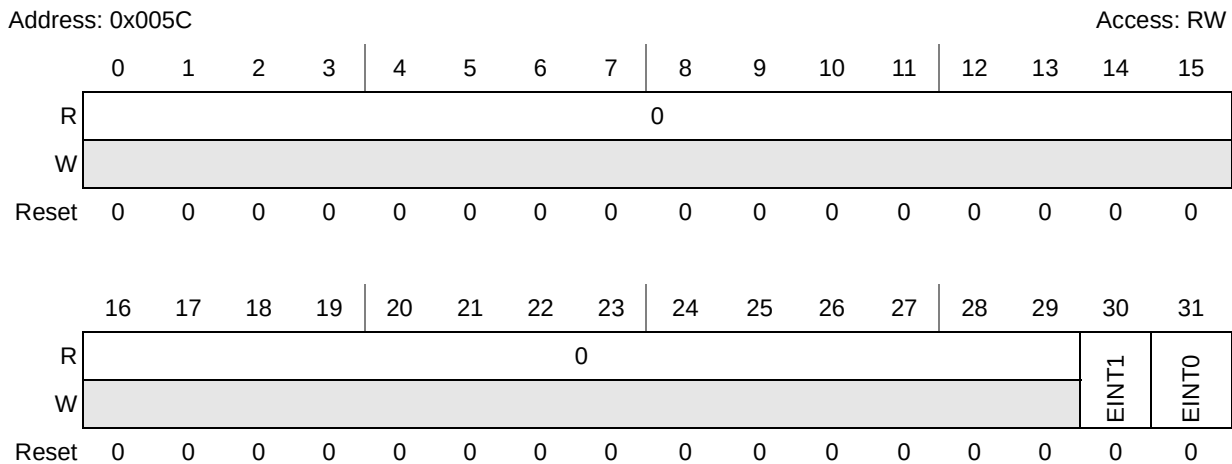


Table 701. Interrupt line enable register field descriptions

Field	Description
0:29	Reserved
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

46.4.5.2.18 Global Filter Configuration Register (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in [Figure 674](#) and [Figure 675](#).

Figure 619. Global filter configuration register

Address: 0x0080														Access: RP			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0																
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										ANFS ⁽¹⁾		ANFE ¹		RRFS ¹	RRFE ¹
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

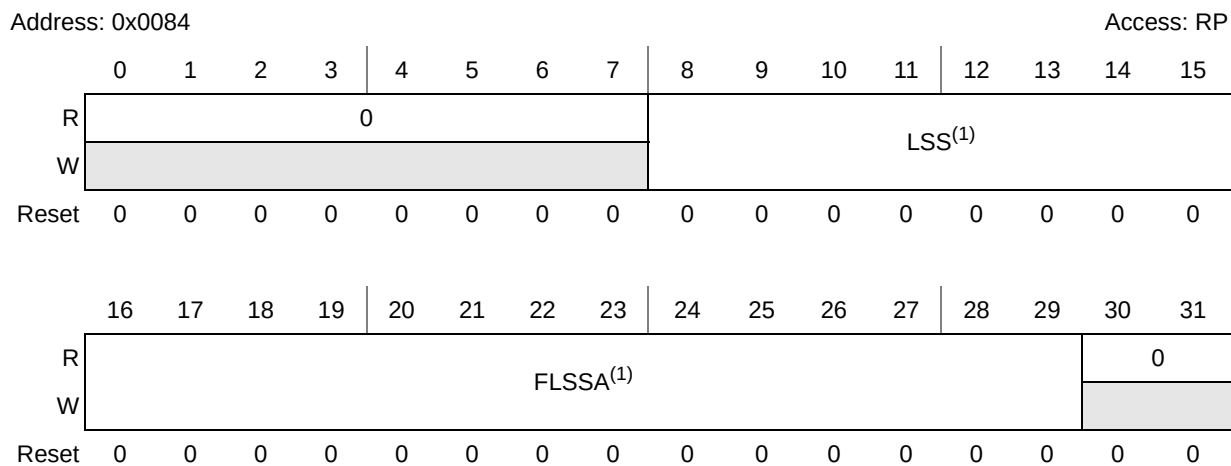
Table 702. Global filter configuration register field descriptions

Field	Description
0:25	Reserved
26:27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28:29 ANFE	Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

46.4.5.2.19 Standard ID Filter Configuration Register (SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in [Figure 674](#)

Figure 620. Standard ID Filter Configuration register



1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

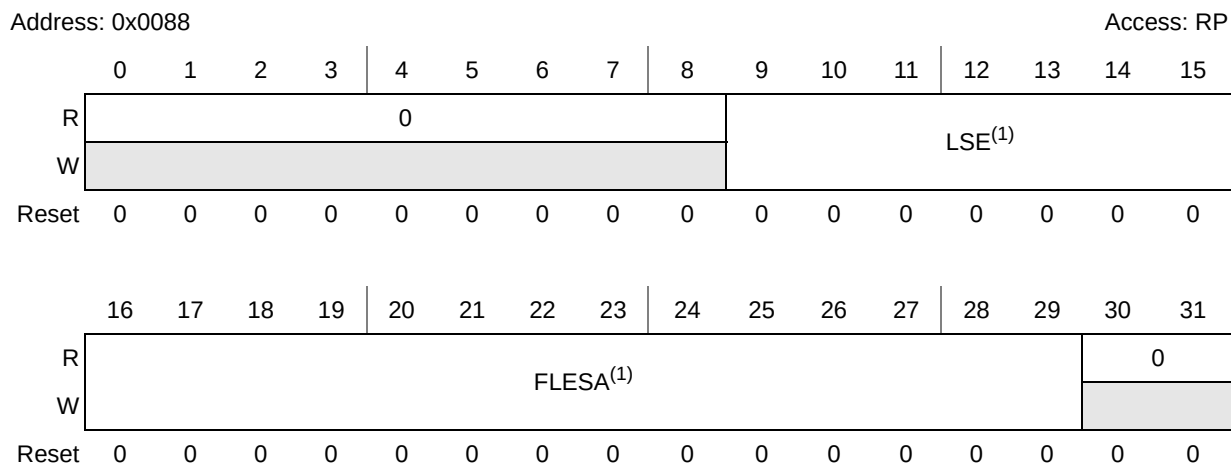
Table 703. Standard ID Filter Configuration register field descriptions

Field	Description
0:7	Reserved
8:15 LSS	List Size Standard 0 No standard Message ID filter 1-128 Number of standard Message ID filter elements >128 Values greater than 128 are interpreted as 128
16:29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, see Figure 668).
30:31	Reserved

46.4.5.2.20 Extended ID Filter Configuration Register (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in [Figure 675](#)

Figure 621. Extended ID Filter Configuration register



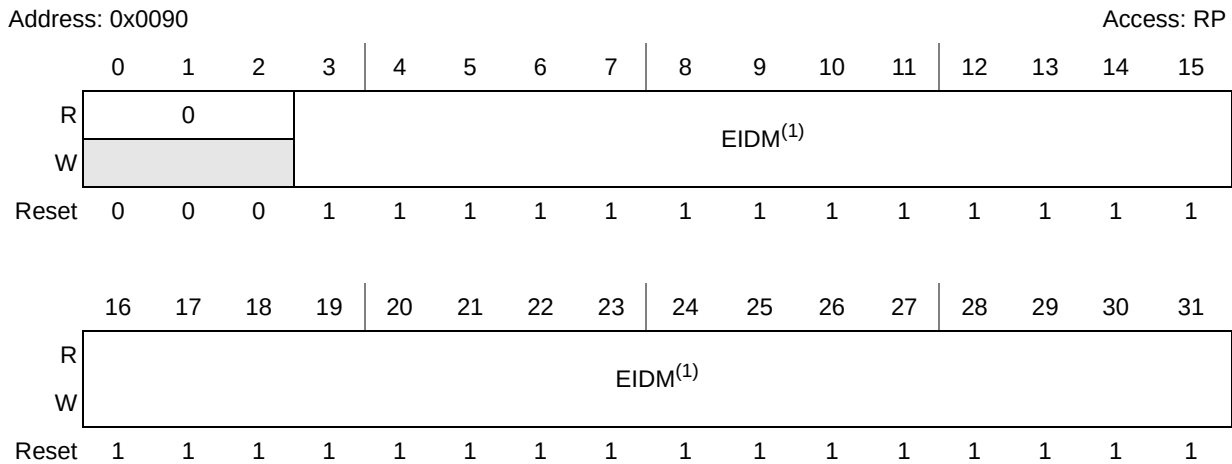
1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 704. Extended ID Filter Configuration register field descriptions

Field	Description
0:8	Reserved
9:15 LSELS	List Size Extended 0 No extended Message ID filter 1-64 Number of extended Message ID filter elements >64 Values greater than 64 are interpreted as 64
16:29 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, see Figure 669).
30:31	Reserved

46.4.5.2.21 Extended ID and Mask Register (XIDAM)

Figure 622. Extended ID and Mask register



1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 705. Extended ID and Mask register field descriptions

Field	Description
0:2	Reserved
3:31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active

46.4.5.2.22 High Priority Message Status Register (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Figure 623. High Priority Message Status register

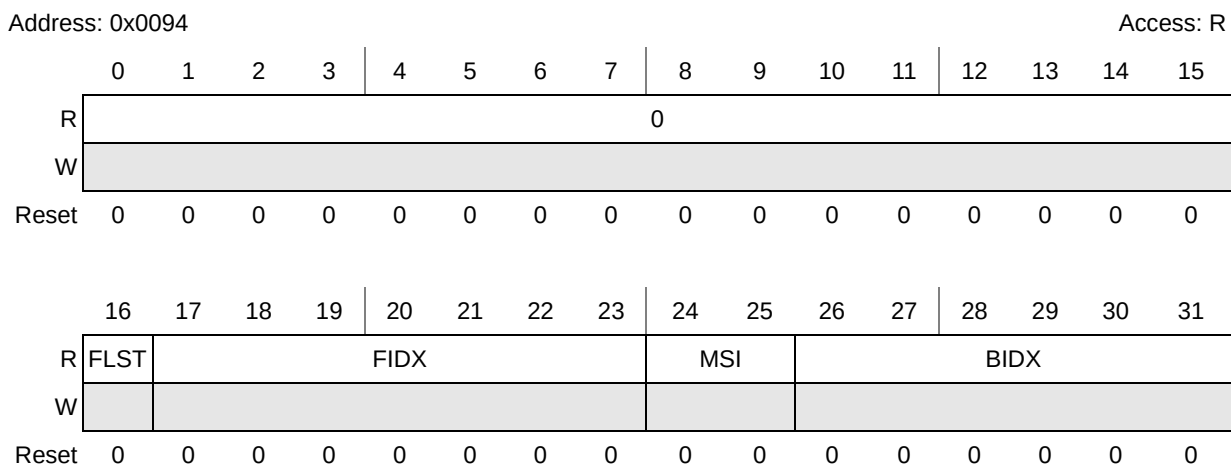


Table 706. High Priority Message Status register field descriptions

Field	Description
0:15	Reserved
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17:23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24:25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26:31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = '1'.

46.4.5.2.23 New Data 1 Register (NDAT1)

Figure 624. NDAT1 register

Address: 0x0098

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 625. NDAT1 register field descriptions

Field	Description
0 ND	New Data[31:0] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

46.4.5.2.24 New Data 2 Register (NDAT2)

Figure 626. NDAT2 register

Address: 0x009C

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32	ND31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 627. NDAT2 register field descriptions

Field	Description
0 ND	New Data[63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

46.4.5.2.25 Rx FIFO 0 Configuration Register (RXF0C)

Figure 628. Rx FIFO 0 Configuration register

Address: 0x00A0

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F0WM ⁽¹⁾							0	F0S ⁽¹⁾						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F0SA ⁽¹⁾														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 707. Rx FIFO 0 Configuration register field descriptions

Field	Description
Bit 0	Reserved
1:7 F0WM	Rx FIFO 0 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) >64 Watermark interrupt disabled
8	Reserved
9:15 F0S	Rx FIFO 0 Size 0 No Rx FIFO 0 1-64 Number of Rx FIFO 0 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1

Table 707. Rx FIFO 0 Configuration register field descriptions (continued)

Field	Description
16:29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, Figure 664).
30:31	Reserved

46.4.5.2.26 Rx FIFO 0 Status Register (RXF0S)

Figure 629. Rx FIFO 0 Status register

Address: 0x0A4

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						RF0L	F0F		0			F0PI			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				F0GI				0				F0FL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 708. Rx FIFO 0 Status register field descriptions

Field	Description
0:5	Reserved
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L] . When IR[RF0L] is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
8:9	Reserved
10:15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.
16:17	Reserved
18:23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
24	Reserved
25:31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

46.4.5.2.27 Rx FIFO 0 Acknowledge Register (RXF0A)

Figure 630. Rx FIFO 0 Acknowledge register

Address: 0x0A8

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										F0AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 709. Rx FIFO 0 Acknowledge register field descriptions

Field	Description
0:25	Reserved
26:31 F0AI	Rx FIFO 0 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI . This will set the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL] .

46.4.5.2.28 Rx Buffer Configuration Register (RXBC)

Figure 631. Rx FIFO 0 Configuration register

Address: 0x00AC

Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RBSA ⁽¹⁾														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 710. Rx FIFO 0 Configuration register field descriptions

Field	Description
0:15	Reserved
16:29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.
30:31	Reserved

46.4.5.2.29 Rx FIFO 1 Configuration Register (RXF1C)

Figure 632. Rx FIFO 1 Configuration register

Address: 0x0B0

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F1WM ⁽¹⁾							0	F1S ⁽¹⁾						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F1SA ⁽¹⁾														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 711. Rx FIFO 1 Configuration register field descriptions

Field	Description
Bit 0	Reserved
Bits 1:7 F1WM	Rx FIFO 1 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 1 watermark interrupt (IR[RF1W]) >64 Watermark interrupt disabled
Bit 8	Reserved
Bits 9:15 F1S	Rx FIFO 1 Size 0 No Rx FIFO 1 1-64 Number of Rx FIFO 1 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1
Bits 16:29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, see Figure 664).

46.4.5.2.30 Rx FIFO 1 Status Register (RXF1S)

Figure 633. Rx FIFO 1 Status register

Address: 0x0B4

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS						RF1L	F1F	0		F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 712. Rx FIFO 1 Status register field descriptions

Field	Description
0:12 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
2:5	Reserved
6 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag IR[RF1L] . When IR[RF1L] is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
8:9	Reserved
10:15 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
16:17	Reserved
18:23 F1GI	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
24	Reserved
25:31 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

46.4.5.2.31 Rx FIFO 1 Acknowledge Register (RXF1A)

Figure 634. Rx FIFO 1 Acknowledge register

Address: 0x0B8																Access: R/W			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											F1A				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 713. Rx FIFO 1 Acknowledge register field description

Field	Description
0:25	Reserved
26:31 F1A	Rx FIFO 1 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI . This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL] .

46.4.5.2.32 Tx Buffer Configuration Register (TXBC)

Figure 635. Tx Buffer Configuration register

Address: 0x0C0																Access: RP			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	TFQM ⁽¹⁾		TFQS ⁽¹⁾						0		NDTB ⁽¹⁾							
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	TBSA ⁽¹⁾														0				
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 714. Tx Buffer Configuration register field description

Field	Description
0	Reserved
1 TFQM	Tx FIFO/Queue Mode. 0 Tx FIFO operation 1 Tx Queue operation
[2:7] TFQS	Transmit FIFO/Queue Size. 0 No Tx FIFO/Queue 1-32 Number of Tx Buffers used for Tx FIFO/Queue >32 Values greater than 32 are interpreted as 32
[8:9]	Reserved
[10:15] NDTB	Number of Dedicated Transmit Buffers. 0 No Dedicated Tx Buffers 1-32 Number of Dedicated Tx Buffers >32 Values greater than 32 are interpreted as 32
[16:29] TBSA	Tx Buffers Start Address. Start address of Tx Buffers section in Message RAM (32-bit word address, Figure 664).
[30:31]	Reserved

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

46.4.5.2.33 Tx FIFO/Queue Status Register (TXFQS)

The Tx FIFO/Queue status is related to the pending Tx requests listed in register **TXBRP**. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (**TXBRP** not yet updated).

Figure 636. Tx FIFO/Queue Status register

Address: 0xC4

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0										TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				TFGI				0				TFFL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 715. Tx FIFO/Queue Status register field description

Field	Description
[0:9]	Reserved
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
[11:15] TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31
[16:18]	Reserved
[19:23] TFGI	Tx FIFO Get Index. Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC.TFQM = '1')
[24:25]	Reserved
[26:31] TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from TFGI , range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = '1')

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Index indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.

For example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

46.4.5.2.34 Tx Buffer Request Pending Register (TXBRP)

Figure 637. Tx Buffer Request Pending register

Address: 0xCC

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TRP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 716. Tx Buffer Request Pending register field description

Field	Description
[0:31] TRP	Transmission Request Pending. Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (see Section 46.4.7.15, Tx handling) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF • after successful transmission together with the corresponding TXBTO bit • when the transmission has not yet been started at the point of cancellation • when the transmission has been aborted due to lost arbitration • when an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

Note: *TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.*

46.4.5.2.35 Tx Buffer Add Request Register (TXBAR)

Figure 638. Tx Buffer Add Request register

Address: 0x0D0																Access: R/W			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	AR																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	AR																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 717. Tx Buffer and Request register field description

Field	Description
[0:31] AR	Add Request. Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

46.4.5.2.36 Tx Buffer Cancellation Request Register (TXBCR)

Figure 639. Tx Buffer Cancellation Request register

Address: 0x0D4

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 718. Tx Buffer Cancellation Request register field description

Field	Description
[0:31] CR	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

46.4.5.2.37 Tx Buffer Transmission Occurred Register (TXBTO)

Figure 640. Tx Buffer Transmission Occurred register

Address: 0x0D8																Access: R															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	TO																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	TO																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

Table 719. Tx Buffer Transmission Occurred register field description

Field	Description
[0:31] TO	Transmission Occurred. Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

46.4.5.2.38 Tx Buffer Cancellation Finished Register (TXBCF)

Figure 641. Tx Buffer Cancellation Finished register

Address: 0x0DC																Access: R															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	CF																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	CF																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

Table 720. Tx Buffer Cancellation Finished register field description

Field	Description
[0:31] CF	Cancellation Finished Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

46.4.5.2.39 Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Figure 642. Tx Buffer Transmission Interrupt Enable register

Address: 0x0E0

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 721. Tx Buffer Transmission Interrupt Enable register field description

Field	Description
[0:31] TIE	Transmission Interrupt Enable Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

46.4.5.2.40 Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Figure 643. Tx Buffer Cancellation Finished Interrupt Enable register

Address: 0x0E4

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 722. Tx Buffer Cancellation Finished Interrupt Enable register field description

Field	Description
[0:31] CFIE	Cancellation Finished Interrupt Enable. Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

46.4.5.2.41 Tx Event FIFO Configuration Register (TXEFC)

Figure 644. Tx Event FIFO Configuration register

Address: 0x0F0

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	EFWM ⁽¹⁾							0	EFS ⁽¹⁾						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EFSA ⁽¹⁾															0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 723. Tx Event FIFO Configuration register field description

Field	Description
[0:1]	Reserved.
[2:7] EFWM	Event FIFO Watermark 0 Watermark interrupt disabled 1-32 Level for Tx Event FIFO watermark interrupt (IR[TEFW]) >32 Watermark interrupt disabled
[8:9]	Reserved.
[10:15] EFS	Event FIFO Size. 0 Tx Event FIFO disabled 1-32 Number of Tx Event FIFO elements >32 Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS -1
[16:29] EFSA	Event FIFO Start Address Start address of Tx Event FIFO in Message RAM (32-bit word address, Figure 664).
[30:31]	Reserved.

46.4.5.2.42 Tx Event FIFO Status Register (TXEFS)

Figure 645. Tx Event FIFO Status register

Address: 0x0F4

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				TEFL		EFF		0		EFPI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		EFGI						0		EFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 724. Tx Event FIFO Status register field description

Field	Description
[0:5]	Reserved.
6 TEFL	Tx Event FIFO Element Lost. This bit is a copy of interrupt flag IR[TEFL] . When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
7 EFF	Event FIFO Full. 0 Tx Event FIFO not full 1 Tx Event FIFO full

Table 724. Tx Event FIFO Status register field description

Field	Description
[8:10]	Reserved.
[11:15] EFPI	Event FIFO Put Index. Tx Event FIFO write index pointer, range 0 to 31.
[16:18]	Reserved
[19:23] EFGI	Event FIFO Get Index. Tx Event FIFO read index pointer, range 0 to 31.
[24:25]	Reserved.
[26:31] EFFL	Event FIFO Fill Level. Number of elements stored in Tx Event FIFO, range 0 to 32.

46.4.5.2.43 Tx Event FIFO Acknowledge Register (TXEFA)

Figure 646. Tx Event FIFO Acknowledge register

Address: 0x0F8												Access: R/W				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												EFAI			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 725. Tx Event FIFO Acknowledge register field description

Field	Description
[0:26]	Reserved.
[27:31] EFAI	Event FIFO Acknowledge Index. After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI . This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL] .

46.4.5.2.44 TT Trigger Memory Configuration Register (TTTMC)

Figure 647. TT Trigger Memory Configuration register

Address: 0x100

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								TME ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TMSA ⁽¹⁾														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 726. TT Trigger Memory Configuration register field description

Field	Description
[0:8]	Reserved.
[9:15] TME	Trigger Memory Elements. 0 No Trigger Memory 1-64 Number of Trigger Memory elements >64 Values greater than 64 are interpreted as 64
[16:29] TMSA	Trigger Memory Start Address. Start address of Trigger Memory in Message RAM (32-bit word address, Figure 664).
[30:31]	Reserved.

46.4.5.2.45 TT Reference Message Configuration Register (TTRMC)

Figure 648. TT Reference Message Configuration register

Address: 0x104

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RMPS (1)	XTD (1)	0	RID ⁽¹⁾												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RID ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 727. TT Reference Message Configuration register field description

Field	Description
0 RMPS	Reference Message Payload Select. Ignored in case of time slaves. 0 Reference message has no additional payload 1 The following elements are taken from Tx Buffer 0: Message Marker MM , Event FIFO Control EFC , Data Length Code DLC , Data Bytes DB (Level 1: bytes 2-8, Level 0, 2: bytes 5-8)
1 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
2	Reserved.
[3:31] RID	Reference Identifier. Identifier transmitted with Reference message and used for Reference message filtering. Standard or extended reference identifier depending on bit XTD . A standard identifier has to be written to ID[28:18] .

46.4.5.2.46 TT Operation Configuration Register (TTOCF)

Figure 649. TT Operation Configuration register

Address: 0x108

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				EVTP	ECC ⁽¹⁾	EGTF ⁽¹⁾	AWL ⁽¹⁾								
W																
Reset	0	0	0	0	0	0	0	0	0x01							

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EECS ⁽¹⁾	IRTO ⁽¹⁾								LDSDL ⁽¹⁾			TM ⁽¹⁾	GEN ⁽¹⁾	0	OM ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 728. TT operation Configuration register field description

Field	Description
[0:4]	Reserved.
5 EVTP	Event Trigger Polarity. 0 Rising edge trigger 1 Falling edge trigger
6 ECC	Enable Clock Calibration. 0 Automatic clock calibration in M_TTCAN Level 0, 2 is disabled 1 Automatic clock calibration in M_TTCAN Level 0, 2 is enabled
7 EGTF	Enable Global Time Filtering. 0 Global time filtering in M_TTCANM_TTCAN Level 0, 2 is disabled 1 Global time filtering in M_TTCAN Level 0, 2 is enabled
[8:15] AWL	Application Watchdog Limit. The application watchdog can be disabled by programming AWL to 0x00. 0x00–FF: Maximum time after which the application has to serve the application watchdog. The application watchdog is incremented once each 256 NTUs.
16 EECS	Enable External Clock Synchronization If enabled, TUR configuration (TURCF[NCL] only) may be updated during M_TTCAN operation. 0 External clock synchronization in M_TTCAN Level 0,2 disabled 1 External clock synchronization in M_TTCAN Level 0,2 enabled
[17:23] IRTO	Initial Reference Trigger Offset. 0x00–7F Positive offset, range from 0 to 127

Table 728. TT operation Configuration register field description

Field	Description
[24:26] LSDSL	LD of Synchronization Deviation Limit. The Synchronization Deviation Limit SDL is configured by its dual logarithm LDSDL with $SDL = 2^{(LDSDL + 5)}$. It should not exceed the clock tolerance given by the CAN bit timing configuration. 0x0–7 LD of Synchronization Deviation Limit ($SDL \leq 32 \dots 4096$)
27 TM	Time Master. 0 Time Master function disabled 1 Potential Time Master
28 GEN	Gap Enable. 0 Strictly time-triggered operation 1 External event-synchronized time-triggered operation
29	Reserved.
[30:31] OM	Operation Mode. 00 Event-driven CAN communication, default 01 M_TTCAN level 1 10 M_TTCAN level 2 11 M_TTCAN level 0

46.4.5.2.47 TT Matrix Limits Register (TTMLM)

Figure 650. TT Matrix Limits register

Address: 0x10C

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				ENTT ⁽¹⁾											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				TXEW ⁽¹⁾				CSS1		CCM ⁽¹⁾					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 729. TT Matrix Limits register field description

Field	Description
[0:3]	Reserved.
[4:15] ENTT	Expected Number of Tx Triggers 0x000–FFF Expected number of Tx Triggers in one Matrix Cycle
[16:19]	Reserved.

Table 729. TT Matrix Limits register field description

Field	Description
[20:23] TXEW	Tx Enable Window. 0x0–F Length of Tx enable window, 1-16 NTU cycles
[24:25] CSS	Cycle Start Synchronization Enables sync pulse output at pin m_ttcan_soc. 00 No sync pulse 01 Sync pulse at start of basic cycle 10 Sync pulse at start of matrix cycle 11 Reserved
[26:31] CCM	Cycle Count Max. 0x00 1 Basic Cycle per Matrix Cycle 0x01 2 Basic Cycles per Matrix Cycle 0x03 4 Basic Cycles per Matrix Cycle 0x07 8 Basic Cycles per Matrix Cycle 0x0F 16 Basic Cycles per Matrix Cycle 0x1F 32 Basic Cycles per Matrix Cycle 0x3F 64 Basic Cycles per Matrix Cycle others Reserved

Note: ISO 11898-4, Section 5.2.1 requires, that only the listed cycle count values are configured. Other values are possible but may lead to inconsistent matrix cycles.

46.4.5.2.48 TUR Configuration Register (TURCF)

The length of the NTU is given by: $NTU = \text{CAN Clock Period} \times \text{NC/DC}$

NC is an 18-bit value. Its high part, **NCH**[17:16] is hard wired to 0b01. Therefore the range of **NC** is 0x10000...0x1FFFF. The value configured by **NCL** is the initial value for **TURN**A[NAV[15:0]]. **DC** is set to 0x1000 by hardware reset and it may not be written to 0x0000.

Level 1: **NC** $\geq 4 \times \text{DC}$ and $NTU = \text{CAN bit time}$

Level 0, 2: **NC** $\geq 8 \times \text{DC}$

The actual value of TUR may be changed by the clock drift compensation function of M_TTCAN Level 0 and Level 2 in order to adjust the node's local view of the NTU to the time master's view of the NTU. **DC** will not be changed by the automatic drift compensation, **TURN**A[NAV] may be adjusted around **NC** in the range of the Synchronization Deviation Limit given by **TTOCF**[LDSDL]. **NC** and **DC** should be programmed to the largest suitable values in order to allow the best computational accuracy for the drift compensation process.

Figure 651. TUR Configuration register

Address: 0x110

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELT ⁽¹⁾	0	DC ⁽¹⁾													
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCL ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 730. TUR Configuration register field description

Field	Description
0 ELT	Enable Local Time. 0 Local time is stopped, default 1 Local time is enabled Please note that the local time is started by setting ELT. It remains active until ELT is reset or until the next hardware reset. TURCF[DC] is locked when TURCF[ELT] = '1'. If ELT is written to '0', the readable value will stay at '1' until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked.
1	Reserved.
[2:15] DC	Denominator Configuration. 0x0000 Illegal value 0x0001–3FFF Denominator Configuration
[16:31] NCL	Numerator Configuration Low. Write access to the TUR Numerator Configuration Low is only possible during configuration with TURCF[ELT] = '0' or if TTOCF[EECS] (external clock synchronization enabled) is set. When a new value for NCL is written outside TT Configuration Mode, the new value takes effect when TTOST.WECS is cleared to '0'. NCL is locked TTOST[WECS] is '1'. 0x0000–FFFF Numerator Configuration Low

Note: If $NC < 7 \times DC$ in *M_TTCAN* Level 1, then it is required that subsequent Time Marks in the Trigger Memory must differ by at least two NTU.

46.4.5.2.49 TT Operation Control Register (TTOCN)

Figure 652. TT Operation Control register

Address: 0x114

Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LCKC	0	ESCN ⁽¹⁾	NIG ⁽¹⁾	TMG ⁽¹⁾	FGP ⁽¹⁾	GCS ⁽¹⁾	TTIE	TMC ⁽¹⁾	RTIE ⁽¹⁾	SWS ⁽¹⁾	SWP ⁽¹⁾	ECS ⁽¹⁾	SGT ⁽¹⁾		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to "1".

Table 731. TT Operation Control register field description

Field	Description
[0:15]	Reserved.
16 LCKC	TT Operation Control Register Locked. Set by a write access to register TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain. 0 Write access to TTOCN enabled 1 Write access to TTOCN locked
17	Reserved.
18 ESCN	External Synchronization Control If enabled the M_TTCAN synchronizes its cycle time phase to an external event signaled by a rising edge at Event Trigger pin(Section 46.4.19, Synchronization to external time schedule). 0 External synchronization disabled 1 External synchronization enabled
19 NIG	Next is Gap. This bit can only be set when the M_TTCAN is the actual Time Master and when it is configured for external event-synchronized time-triggered operation (TTOCF[GEN] = '1') 0 No action, reset by reception of any Reference message 1 Transmit next Reference Message with Next_is_Gap = '1'
20 TMG	Time Mark Gap. 0 Reset by each Reference message 1 Next Reference message started when Register Time Mark interrupt TTIR[RTMI] is activated

Table 731. TT Operation Control register field description

Field	Description
21 FGP	Finish Gap. Set by the CPU, reset by each reference message 0 No reference message requested 1 Application requested start of reference message
22 GCS	Gap Control Select 0 Gap control independent from Event trigger 1 Gap control by input Event trigger pin
23 TTIE	Trigger Time Mark Interrupt Pulse Enable External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Schedule or In_Gap. 0 Trigger Time Mark Interrupt output m_ttcn_tmp disabled 1 Trigger Time Mark Interrupt output m_ttcn_tmp enabled
24:25 TMC	Register Time Mark Compare. 00 No Register Time Mark Interrupt generated 01 Register Time Mark Interrupt if Time Mark = cycle time 10 Register Time Mark Interrupt if Time Mark = local time 11 Register Time Mark Interrupt if Time Mark = global time Please note when changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = "00", then reconfigure TTTMK, and finally set TMC to the intended time reference.
26 RTIE	Register Time Mark Interrupt Pulse Enable. Register time mark interrupts are configured by register TTTMK . A register time mark interrupt pulse with the length of one m_ttcn_clk period is generated when time referenced by TTOCN[TMC] (cycle, local, or global) equals TTTMK[TM] , independent of the synchronization state. 0 Register Time Mark Interrupt output disabled 1 Register Time Mark Interrupt output enabled
[27:28] SWS	Stop Watch Source. 00 Stop Watch disabled 01 Actual value of cycle time is copied to TTCPT[SWV] 10 Actual value of local time is copied to TTCPT[SWV] 11 Actual value of global time is copied to TTCPT[SWV]
29 SWP	Stop Watch Polarity. 0 Rising edge trigger 1 Falling edge trigger
30 ECS	External Clock Synchronization. Writing a '1' to ECS sets TTOST[WECS] if the node is the actual Time Master. ECS is reset after one Host clock period. The external clock synchronization takes effect at the start of the next basic cycle.
31 SGT	Set Global time. Writing a '1' to SGT sets TTOST[WGDT] if the node is the actual Time Master. SGT is reset after one Host clock period. The global time preset takes effect when the node transmits the next Reference message with the Master_Ref_Mark modified by the preset value written to TTGTP .

46.4.5.2.50 TT Global Time Preset Register (TTGTP)

If **TTOST.WGDT** is set, the next Reference message will be transmitted with the **Master_Ref_Mark** modified by the preset value and with **Disc_Bit** = '1', presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a Reference message with **Disc_Bit** = '1' becomes valid or if the node is not the current Time Master. **TP** is locked while **TTOST[WGTD]** = '1' after setting **TTOCN[SGT]** until the Reference message with **Disc_Bit** = '1' becomes valid or until the node is no longer the current Time Master.

Table 0-2.

Address: 0x118																Access: R/W			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	CTP																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	TP																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Figure 653. TT Global Time Preset register

Table 732. TT Global Time Preset register field description

Field	Description
[0:15] CTP	Cycle Time Target Phase. CTP is write-protected while TTOCN[ESCN] or TTOST[SPL] are set (see Section 46.4.19, Synchronization to external time schedule). 0x0000–FFFF Defines target value of cycle time when a rising edge of event trigger is expected
[16:31] TP	Time Preset. TP is write-protected while TTOST[WGTD] is set. 0x0000–7FFF Next Master Reference Mark = Master Reference Mark + TP 0x8000 reserved 0x8001–FFFF Next Master Reference Mark = Master Reference Mark - (0x10000 - TP)

46.4.5.2.51 TT Time Mark Register (TTTMK)

A time mark interrupt (**TTIR[TMI]** = '1') is generated when the time base indicated by **TTOCN[TMC]** (cycle time, local time, or global time) has the same value as **TM**.

Figure 654. TT Time Mark register

Address: 0x11C																Access: R/W															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	LCKM	0							TICC																						
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	TM																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

Table 733. TT Time Mark register field description

Field	Description
0 LCKM	TT Time Mark Register Locked. Always set by a write access to registers TTOCN. Set by write access to register TTTMK when TTOCN[TMC] ≠ "00". Reset when the registers have been synchronized into the CAN clock domain. 0 Write access to TTTMK enabled 1 Write access to TTTMK locked
1:8	Reserved
[9:15] TICC	Time Mark Cycle Code. Cycle count for which the time mark is valid. 0b000000x valid for all cycles 0b000001c valid every second cycle at cycle count mod2 = c 0b00001cc valid every fourth cycle at cycle count mod4 = cc 0b0001ccc valid every eighth cycle at cycle count mod8 = ccc 0b001cccc valid every sixteenth cycle at cycle count mod16 = cccc 0b01ccccc valid every thirty-second cycle at cycle count mod32 = cccccc 0b1cccccc valid every sixty-fourth cycle at cycle count mod64 = ccccccc
[16:31] TM	Time Mark. 0x0000–FFFF Time Mark

Note: When using byte access to register TTTMK it is recommended to first disable the time mark compare function (**TTOCN[TMC]** = "00") to avoid compares on inconsistent register values.

46.4.5.2.52 TT Interrupt Register (TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.

Figure 655. TT Interrupt Register

Address: 0x120

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												CER	AW	WT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWT	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 734. TT Interrupt Register field description

Field	Description
[0:12]	Reserved.
13 CER	Configuration Error. Trigger out of order. 0 No error found in trigger list 1 Error found in trigger list
14 AW	Application Watchdog. 0 Application watchdog served in time 1 Application watchdog not served in time
15 WT	Watch Trigger. 0 No missing Reference message 1 Missing Reference message (Level 0: cycle time 0xFF00)
16 IWT	Initialization Watch Trigger. The initialization is restarted by resetting IWT. 0 No missing Reference message during system startup 1 No system startup due to missing Reference message
17 ELC	Error Level Changed. Not set when error level changed during initialization. 0 No change in error level 1 Error level changed
18 SE2	Scheduling Error 2. 0 No scheduling error 2 1 Scheduling error 2 occurred

Table 734. TT Interrupt Register field description

Field	Description
19 SE1	Scheduling Error 1. 0 No scheduling error 1 1 Scheduling error 1 occurred
20 TXO	Tx Count Overflow. 0 Number of Tx Trigger as expected 1 More Tx trigger than expected in one cycle
21 TXU	Tx Count Underflow. 0 Number of Tx Trigger as expected 1 Less Tx trigger than expected in one cycle
22 GTE	Global Time Error. Synchronization deviation SD exceeds limit specified by TTOCF[LDSDL] , M_TTCAN Level 0, 2 only. 0 Synchronization deviation within limit 1 Synchronization deviation exceeded limit
23 GTD	Global Time Discontinuity. 0 No discontinuity of global time 1 Discontinuity of global time
24 GTW	Global Time Wrap. 0 No global time wrap occurred 1 Global time wrap from 0xFFFF to 0x0000 occurred
25 SWE	Stop Watch Event. 0 No rising edge at stop watch trigger pin detected 1 Rising edge at stop watch trigger pin detected
26 TTMI	Trigger Time Mark Event Internal. Internal time mark events are configured by trigger memory element TMIN (see Section 46.4.6.6, Trigger memory element). Set when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Gap or In_Schedule. 0 Time mark not reached 1 Time mark reached (Level 0: cycle time TTOCF[RTO] x 0x200)
27 RTMI	Register Time Mark Interrupt. Set when time referenced by TTOCN[TMC] (cycle, local, or global) equals TTTMK[TM] , independent of the synchronization state. 0 Time mark not reached 1 Time mark reached
28 SOG	Start of Gap. 0 No reference message seen with Next_is_Gap bit set 1 Reference message with Next_is_Gap bit set becomes valid
29 CSM	Change of Synchronization Mode. 0 No change in master to slave relation or schedule synchronization 1 Master to slave relation or schedule synchronization changed, also set when TTOST[SPL] is reset

Table 734. TT Interrupt Register field description

Field	Description
30 SMC	Start of Matrix Cycle. 0 No Matrix Cycle started since bit has been reset 1 Matrix Cycle started
31 SBC	Start of Basic Cycle. 0 No Basic Cycle started since bit has been reset 1 Basic Cycle started

46.4.5.2.53 TT Interrupt Enable Register (TTIE)

The settings in the TT Interrupt Enable register determine which status changes in the TT Interrupt Register will result in an interrupt.

Figure 656. TT Interrupt Enable register

Address: 0x124

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													CERE	AWE	WTE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 735. TT Interrupt Enable register field description

Field	Description
Bits [0:12]	Reserved.
13 CERE	Configuration Error Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
14 AWE	Application Watchdog Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
15 WTE	Watch Trigger Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
16 IWTE	Initialization Watch Trigger Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled

Table 735. TT Interrupt Enable register field description

Field	Description
17 ELCE	Change Error Level Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
18 SE2E	Scheduling Error 2 Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
19 SE1E	Scheduling Error 1 Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
20 TXOE	Tx Count Overflow Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
21 TXUE	Tx Count Underflow Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
22 GTEE	Global Time Error Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
23 GTDE	Global Time Discontinuity Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
24 GTWE	Global Time Wrap Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
25 SWEE	Stop Watch Event Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
26 TTMIE	Trigger Time Mark Event Internal Enable. 0 TT interrupt disabled 1 TT interrupt enabled
27 RTMIE	Register Time Mark Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
28 SOGE	Start of Gap Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
29 CSME	Change of Synchronization Mode Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled

Table 735. TT Interrupt Enable register field description

Field	Description
30 SMCE	Start of Matrix Cycle Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
31 SBCE	Start of Basic Cycle Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled

46.4.5.2.54 TT Interrupt Line Select Register (TTILS)

The TT Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the TT Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1].

0 TT interrupt assigned to interrupt line 0

1 TT interrupt assigned to interrupt line 1

Figure 657. TT Interrupt Line Select register

Address: 0x128

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												CERL	AWL	WTL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 736. TT Interrupt Line Select register field description

Field	Description
[0:12]	Reserved.
13 CERL	Configuration Error Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
14 AWL	Application Watchdog Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
15 WTL	Watch Trigger Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1

Table 736. TT Interrupt Line Select register field description

Field	Description
16 IWTL	Initialization Watch Trigger Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
17 ELCL	Change Error Level Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
18 SE2L	Scheduling Error 2 Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
19 SE1L	Scheduling Error 1 Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
20 TXOL	Tx Count Overflow Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
21 TXUL	Tx Count Underflow Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
22 GTCL	Global Time Error Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
23 GTDL	Global Time Discontinuity Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
24 GTWE	Global Time Wrap Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
25 SWEL	Stop Watch Event Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
26 TTMIL	Trigger Time Mark Event Internal Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
27 RTMIL	Register Time Mark Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
28 SOGI	Start of Gap Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
29 CSML	Change of Synchronization Mode Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1

Table 736. TT Interrupt Line Select register field description

Field	Description
30 SMCL	Start of Matrix Cycle Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
31 SBCL	Start of Basic Cycle Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1

46.4.5.2.55 TT Operation Status Register (TTOST)

Figure 658. TT Operation Status register

Address: 0x12C

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SPL	WECS	AWE	WFE	GSI	TMP			GFI	WGTD	0					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RTO								QCS	QGTP	SYS		MS		EL	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 737. TT Operation Status register field description

Field	Description
0 SPL	Schedule Phase Lock. The bit is valid only when external synchronization is enabled (TTOCN[ESCN] = '1'). In this case it signals that the difference between cycle time configured by TTGTP[CTP] and the cycle time at the rising edge at event trigger pin is less or equal 9 NTU (see Section 46.4.19, Synchronization to external time schedule). 0 Phase outside range 1 Phase inside range
1 WECS	Wait for External Clock Synchronization. 0 No external clock synchronization pending 1 Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.
2 AWE	Application Watchdog Event. The application watchdog is served by reading TTOST. When the watchdog is not served in time, bit AWE is set, all M_TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode. 0 Application Watchdog served in time 1 Failed to serve Application Watchdog in time

Table 737. TT Operation Status register field description

Field	Description
3 WFE	Wait for Event. 0 No Gap announced, reset by a Reference Message with Next_is_Gap = '0' 1 Reference Message with Next_is_Gap = '1' received
4 GSI	Gap Started Indicator. 0 No Gap in schedule, reset by each reference message and for all time slaves 1 Gap time after Basic Cycle has started
[5:7] TMP	Time Master Priority. 0x0-7 Priority of actual Time Master
8 GFI	Gap Finished Indicator. Set when the CPU writes TTOCN[FGP] , or by a Time Mark Interrupt if TMG = '1', or via input pin (event trigger) if TTOCN[GCS] = '1'. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message. 0 Reset at the end of each reference message 1 Gap finished by M_TTCAN
9 WGTD	Wait for Global Time Discontinuity. 0 No global time preset pending 1 Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a Reference message with Disc_Bit = '1' or after it received a Reference message.
[16:23] RTO	Reference Trigger Offset. The Reference Trigger Offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of -127 is reached. In case the M_TTCAN becomes Time Master (MS[1:0] = "11"), the reset of RTO is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by TTOCF[IRTO] is read. 0x00-FF Actual Reference Trigger offset value
24 QCS	Quality of Clock Speed. Only relevant in M_TTCAN Level 0 and Level 2, otherwise fixed to '1'. 0 Local clock speed not synchronized to Time Master clock speed 1 Synchronization Deviation $\leq$ SDL
25 QGTP	Quality of Global Time Phase. Only relevant in M_TTCAN Level 0 and Level 2, otherwise fixed to '0'. 0 Global time not valid 1 Global time in phase with Time Master
[26:27] SYS	Synchronization State. 00 Out of Synchronization 01 Synchronizing to M_TTCAN communication 10 Schedule suspended by Gap (In_Gap) 11 Synchronized to schedule (In_Schedule)

Table 737. TT Operation Status register field description

Field	Description
[28:29] MS	Master State. 00 Master_Off, no master properties relevant 01 Operating as Time Slave 10 Operating as Backup Time Master 11 Operating as current Time Master
[30:31] EL	Error Level. 00 Severity 0—No Error 01 Severity 1—Warning 10 Severity 2—Error 11 Severity 3—Severe Error

46.4.5.2.56 TUR Numerator Actual Register (TURNA)

There is no drift compensation in M_TTCAN Level 1 (**NAV = NC**). In M_TTCAN Level 0 and Level 2, the drift between the node's local clock and the time master's local clock is calculated. The drift is compensated when the Synchronization Deviation (difference between **NC** and the calculated **NAV**) is not more than $2^{(TTOCF[LDSDL] + 5)}$. With **TTOCF[LDSDL] ≤ 7**, this results in a maximum range for **NAV** of $(NC - 0x1000) ≤ NAV ≤ (NC + 0x1000)$.

Figure 659. TUR Numerator Actual register

Address: 0x130																Access: R	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0														NAV		
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	NAV																
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 738. TUR Numerator Actual register field description

Field	Description
[0:13]	Reserved.
[16:31] NAV	Numerator Actual Value. ≤ 0x0EFFF Illegal value 0x0F000–20FFF Actual numerator value ≥ 0x21000 Illegal value

46.4.5.2.57 TT Local and Global Time Register (TTLGT)

Figure 660. TT Local and Global Time register

Address: 0x134 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 739. TT Local and Global Time register field description

Field	Description
[0:15] GT	Global Time. Non-fractional part of the sum of the node's local time and its local offset (see Section 46.4.14, Local time, cycle time, global time, and external clock synchronization). 0x0000–FFFF Global time value of M_TTCAN network
[16:31] LT	Local Time. Non-fractional part of local time, incremented once each local NTU (see Section 46.4.14, Local time, cycle time, global time, and external clock synchronization). 0x0000–FFFF Local time value of M_TTCAN node

46.4.5.2.58 TT Cycle Time and Count Register (TTCTC)

Figure 661. TT Cycle Time and Count register

Address: 0x138 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											CC				
W																
Reset	0	0	0	0	0	0	0	0	0	0			0x3F			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 740. TT Cycle Time and Count register field description

Field	Description
[0:9]	Reserved.
[10:15] CC	Cycle Count. 0x00–3F Number of actual Basic Cycle in the System Matrix
[16:31] CT	Cycle Time. Non-fractional part of the difference of the node's local time and Ref_Mark (see Section 46.4.14, Local time, cycle time, global time, and external clock synchronization). 0x0000–FFFF Cycle time value of M_TTCAN Basic Cycle

46.4.5.2.59 TT Capture Time Register (TTCPT)

Figure 662. TT Capture Time register

Address: 0x13C

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										CCV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 741. TT Capture Time register field description

Field	Description
[0:15] SWV	Stop Watch Value. On a rising/falling edge (as configured via TTOCN[SWP]) at the Stop Watch Trigger pin , when TTOCN[SWS] is $\neq$ "00" and TTIR[SWE] is '0', the actual time value as selected by TTOCN[SWS] (cycle, local, global) is copied to SWV and TTIR[SWE] will be set to '1'. Capturing of the next stop watch value is enabled by resetting TTIR[SWE] . 0x0000–FFFF Captured Stop Watch value
[26:31] CCV	Cycle Count Value. Cycle count value captured together with SWV . 0x00–3F Captured cycle count value

46.4.5.2.60 TT Cycle Sync Mark Register (TTCSM)

Figure 663. TT Cycle Sync Mark register

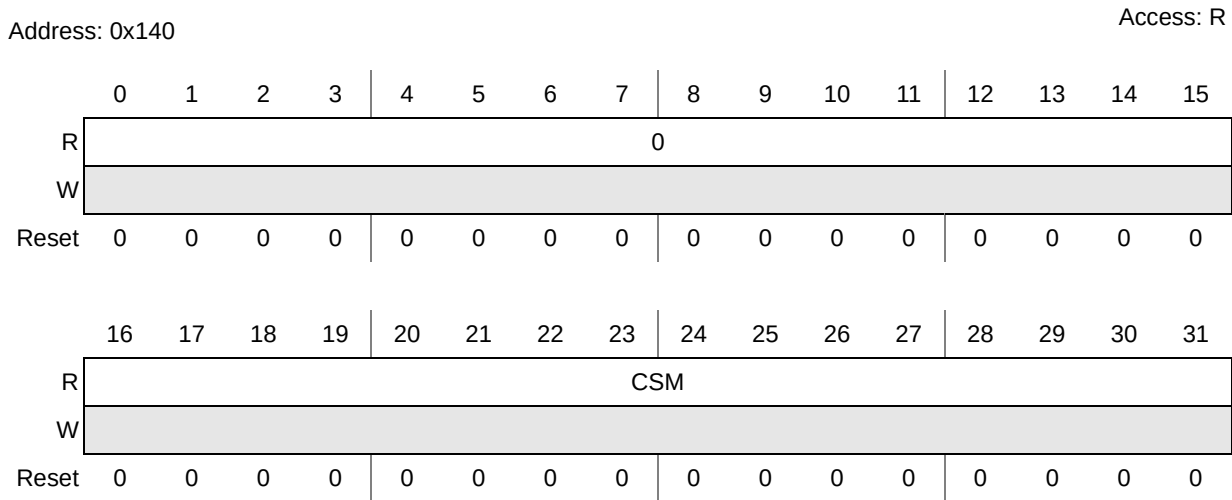


Table 742. TT Cycle Sync Mark register field description

Field	Description
[0:15]	Reserved.
[16:31] CSM	Cycle Sync Mark. The Cycle Sync Mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid. 0x0000–FFFF Captured cycle time

46.4.6 Message RAM

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_TTCAN module can be configured to allocate up to 1344 words in the Message RAM. It is not necessary to configure each of the sections listed in the following figure, nor is there any restriction with respect to the sequence of the sections.

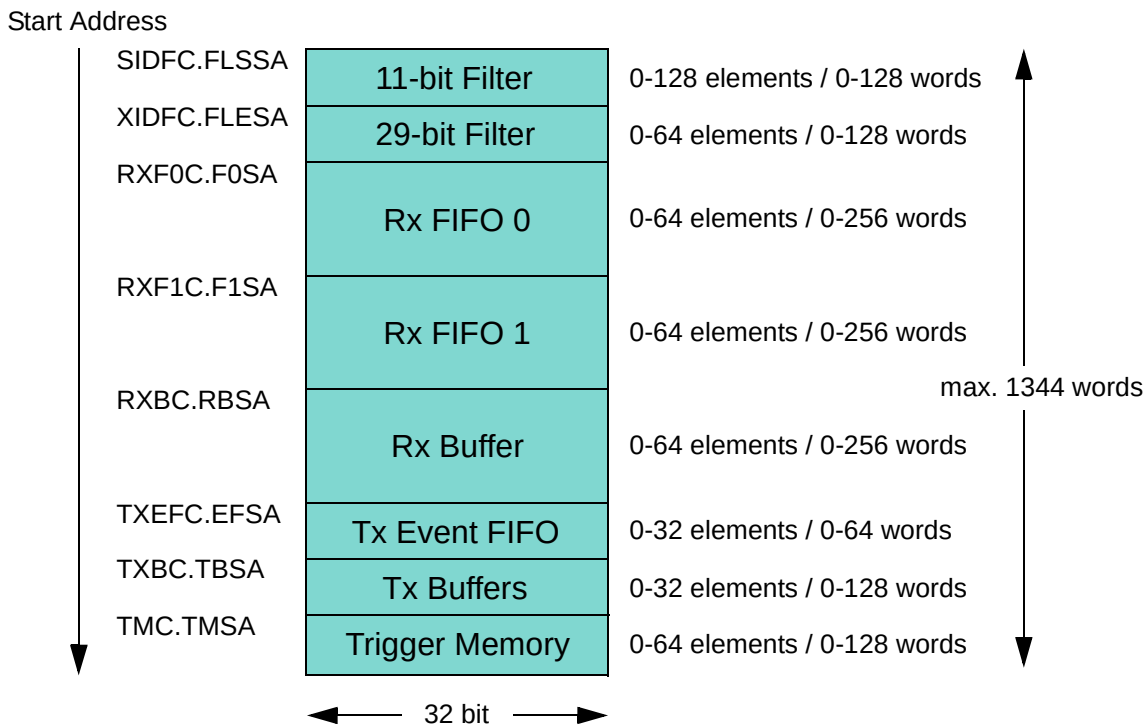


Figure 664. Message RAM Configuration

When the M_TTCAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses.

Note: The M_TTCAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

46.4.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer/ FIFO element is shown below.

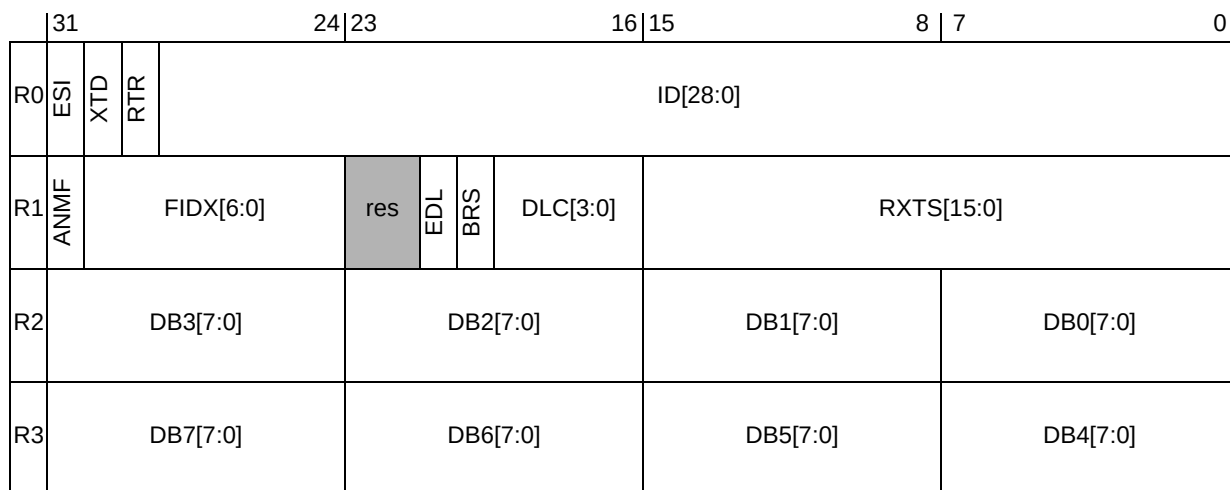


Figure 665. Rx Buffer and FIFO Element

Table 743. Rx Buffer and FIFO Element descriptions

R0 Bit 31 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 Bit 30 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit extended identifier
R0 Bit 29 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame
R0 Bits 28:0 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 Bit 21 EDL Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)

Table 743. Rx Buffer and FIFO Element descriptions (continued)

R1 Bit 20 BRS Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]: Data Length Code	0-8 Received frame has 0-8 data bytes 9-15 Received frame has 8 data bytes
R1 Bits 15:0 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 Bits 31:24	DB3[7:0]: Data Byte 3
R2 Bits 23:16	DB2[7:0]: Data Byte 2
R2 Bits 15:8	DB1[7:0]: Data Byte 1
R2 Bits 7:0	DB0[7:0]: Data Byte 0
R3 Bits 31:24	DB7[7:0]: Data Byte 7
R3 Bits 23:16	DB6[7:0]: Data Byte 6
R3 Bits 15:8	DB5[7:0]: Data Byte 5
R3 Bits 7:0	DB4[7:0]: Data Byte 4

46.4.6.2 Tx Buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC[TFQS] and TXBC[NDTB].

Figure 666. Tx Buffer Element

	31			24	23			16	15			8	7		0
T0	res	XTD	RTR	ID[28:0]											
T1	MM[7:0]			EFC	res	DLC[3:0]	res								
T2	DB3[7:0]			DB2[7:0]			DB1[7:0]			DB0[7:0]					
T3	DB7[7:0]			DB6[7:0]			DB5[7:0]			DB4[7:0]					

Table 744. Tx Buffer Element descriptions

T0 Bit 30 XTD: Extended Identifier	0 11-bit standard identifier 1 29-bit extended identifier
T0 Bit 29 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame
T0 Bits 28:0 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO Control 0 Don't store Tx events 1 Store Tx events
T1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes
T2 Bits 31:24	DB3[7:0]: Data Byte 3
T2 Bits 23:16	DB2[7:0]: Data Byte 2
T2 Bits 15:8	DB1[7:0]: Data Byte 1
T2 Bits 7:0	DB0[7:0]: Data Byte 0
T3 Bits 31:24	DB7[7:0]: Data Byte 7
T3 Bits 23:16	DB6[7:0]: Data Byte 6
T3 Bits 15:8	DB5[7:0]: Data Byte 5
RT3 Bits 7:0	DB4[7:0]: Data Byte 4

46.4.6.3 Tx Event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

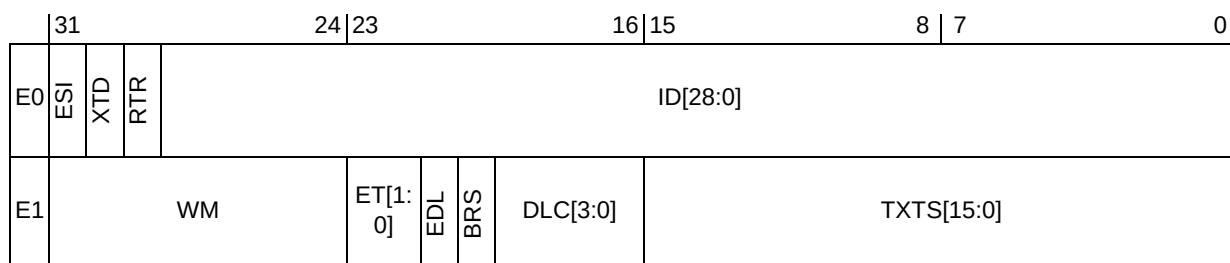


Figure 667. Tx Event FIFO Element

Table 745. Tx Event FIFO Element

E0 Bit 31 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 Bit 30 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 Bit 29 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 Bits 31:24 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status
E1 Bit 23:22 ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 Bit 21 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching
E1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 Frame with 0-8 data bytes transmitted 9-15 Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

46.4.6.4 Standard message ID Filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC.FLSSA plus the index of the filter element (0...127).

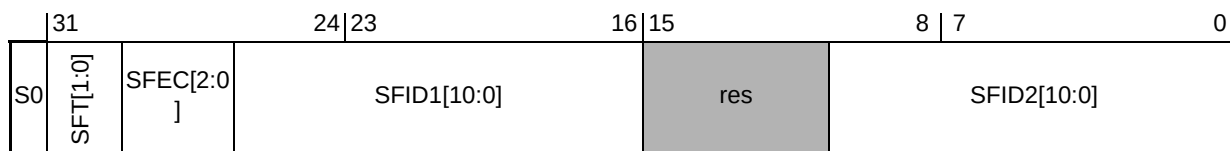


Figure 668. Standard Message ID Filter Element

Table 746. Standard Message ID Filter Element Field Description

Bits 31:30 SFT[1:0]	Standard Filter Type First ID of standard ID filter element.
Bit 29:27 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = "100", "101", or "110" a match sets interrupt flag IE[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer, configuration of SFT[1:0] ignored
Bits 26:16 SFID1[10:0]	Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 10:0 SFID2[10:0]	Standard Filter ID 2 Second ID of standard ID filter element

Note: In case a reserved value is configured, the filter element is considered disabled.

46.4.6.5 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address $XIDFC[FLESA]$ plus two times the index of the filter element (0...63).

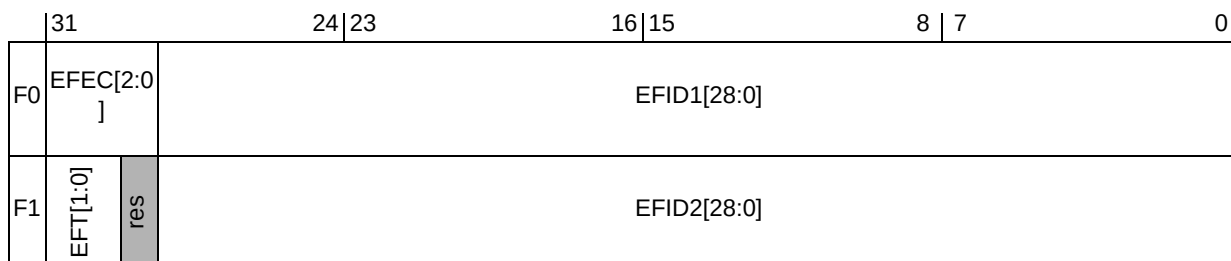


Figure 669. Extended Message ID Filter Element

Table 747. Extended Message ID Filter Element Field Description

F0 Bits 31:29 EFEC[2:0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = "100", "101", or "110" a match sets interrupt flag IE[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer, configuration of EFT[1:0] ignored
F0 Bits 28:0 EFID1[28:0]	Extended Filter ID 1 First ID of extended ID filter element.
F1 Bits 31:30 EFT[1:0]	Extended Filter Type 00 Range filter from EF1ID to EF2ID ($EF2ID \geq EF1ID$) 01 Dual ID filter for EF1ID or EF2ID 10 Classic filter: EF1ID = filter, EF2ID = mask 11 Range filter from EF1ID to EF2ID ($EF2ID \geq EF1ID$), XIDAM mask not applied
F1 Bits 28:0 EFID2[28:0]	Extended Filter ID 2

46.4.6.6 Trigger memory element

Up to 64 trigger memory elements can be configured. When accessing a Trigger Memory element, its address is the Trigger Memory Start Address TTTMC[TMSA] plus the index of the trigger memory element (0...63).

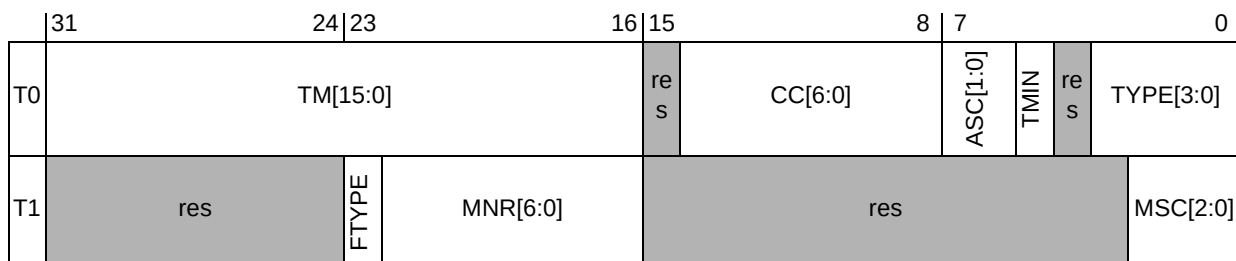


Figure 670. Trigger Memory Element

Table 748. Trigger Memory Element Field Description

T0 Bit 31:16 TM[15:0]	Time Mark Cycle time for which the trigger becomes active.
T0 Bit 14:8 CC[6:0]	Cycle Code Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List. 0b000000x valid for all cycles 0b000001c valid every 2nd cycle at cycle count mod2 = c 0b00001cc valid every 4th cycle at cycle count mod4 = cc 0b0001ccc valid every 8th cycle at cycle count mod8 = ccc 0b001cccc valid every 16th cycle at cycle count mod16 = cccc 0b01ccccc valid every 32nd cycle at cycle count mod32 = cccccc 0b1cccccc valid every 64th cycle at cycle count mod64 = ccccccc
T0 Bit 7:6 ASC[1:0]	Time Mark Event Internal 0 No action 1 TTIR.TTMI is set when trigger memory element becomes active
T0 Bit 5 TMIN	Extended Filter ID 2 Second ID of extended ID filter element.

Table 748. Trigger Memory Element Field Description (continued)

T0 Bit 3:0 TYPE[3:0]:	Trigger Type 0000 Tx_Ref_Trigger - valid when not in Gap 0001 Tx_Ref_Trigger_Gap - valid when in Gap 0010 Tx_Trigger_Single - starts a single transmission in an exclusive time window 0011 Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window 0100 Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window 0101 Tx_Trigger_Merged - starts a merged arbitration window 0110 Watch_Trigger - valid when not in Gap 0111 Watch_Trigger_Gap - valid when in Gap 1000 Rx_Trigger - check for reception 1001 Time_Base_Trigger - only control TMIN, TMEX, and ASC 1010...1111=End_of_List - illegal type, causes config error Note: For ASC operation (ASC = "10", "11") only trigger types Rx_Trigger and Time_Base_Trigger should be used.
T1 Bit 23 FTYPE	Filter Type 0 '11-bit standard message ID 1 29-bit extended message ID
T1 Bit 22:16 MNR[6:0]	Message Number Transmission: Trigger is valid for configured Tx Buffer number. Valid values are 0 to 31. Reception: Trigger is valid for standard / extended message ID filter element number. Valid values are 0 to 63 resp. 0 to 127.
T1 Bits 2:0 MSC[2:0]	Message Status Count Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1). 0-7= Actual status Note: The trigger memory elements have to be written when the M_TTCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual Reference Trigger Offset (TTOST[RTO]).

46.4.7 M_TTCAN functional description

46.4.7.1 Operating modes

46.4.7.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (=Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_TTCAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write). CCCR[CCE] can only be set/reset while CCCR[INIT] = 1. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set:

- HPMS - High Priority Message Status
- RXF0S - Rx FIFO 0 Status
- RXF1S - Rx FIFO 1 Status
- TXFQS - Tx FIFO/Queue Status
- TXBRP - Tx Buffer Request Pending
- TXBTO - Tx Buffer Transmission Occurred
- TXBCF - Tx Buffer Cancellation Finished
- TXEFS - Tx Event FIFO Status
- TTOST - TT Operation Status
- TTLGT - TT Local and Global Time, only Global Time TTLGT[GT] is reset
- TTCTC - TT Cycle Time and Count
- TTCSM - TT Cycle Sync Mark

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = '1'.

The following registers are only writeable while CCCR[CCE] = '0'

- TXBAR - Tx Buffer Add Request
- TXBCR - Tx Buffer Cancellation Request

CCCR[TEST] and CCCR[MON] can only be set by the Host while CCCR[INIT] = '1' and CCCR[CCE] = '1'. Both bits may be reset at any time. CCCR[DAR] can only be set/reset while CCCR[INIT] = '1' and CCCR[CCE] = '1'.

46.4.7.2 Normal operation

The M_TTCAN's default operating mode after hardware reset is event-driven CAN communication without time triggers (TTOCF[OM] = "00"). It is required that both CCCR[INIT] and CCCR[CCE] are set before the TT Operation Mode can be changed.

Once the M_TTCAN is initialized and CCCR[INIT] is reset to zero, the M_TTCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

46.4.7.3 CAN FD Operation

There are two stages in the CAN FD protocol, first the Long Frame Mode where the data field of a CAN frame may be longer than 8 bytes. The second stage is the Fast Frame Mode where control field, data field, and CRC field of a CAN Frame are transmitted with a higher

bit rate than the beginning and the end of the frame. Fast Frame Mode can only be used in combination with Long Frame Mode.

The CAN operation mode is chosen by programming CCCR.CME. In case CCCR.CME = g01h transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CCCR.CME = g10h/g11h transmission and reception of long and fast CAN FD frames is enabled. CCCR.CME can only be changed while CCCR.INIT and CCCR.CCE are both set.

When initialization is left (CCCR.INIT set to e0f), both Long Frame Mode and Fast Frame Mode are inactive, they have to be requested by writing to CCCR.CMR. A mode change requested by writing to CCCR.CMR will be executed next time the CAN protocol

controller FSM reaches idle phase between CAN frames. Upon this event CCCR.CMR is reset to g00h and the status flags CCCR.FACT and CCCR.LACT are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CCCR.CMR is retained until it is overwritten by the next mode change request. Default is normal CAN operation. A change of the CAN operation mode should not be requested while there are pending transmission requests.

When CCCR.CME , g00h, received frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD long frame, EDL = dominant signifies a standard CAN frame. In a CAN FD long frame, the two bits following EDL, r0 and BRS, decide whether this CAN FD long frame is also a CAN FD fast frame.

A CAN FD fast frame is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol. The status bits CCCR.LACT and CCCR.FACT indicate the format of transmitted frames. When CCCR.LACT is set, frames will be transmitted in CAN FD long format with EDL = recessive. When both CCCR.LACT and CCCR.FACT are set, frames will be transmitted in CAN FD Fast format with both EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to the following table:

Table 749. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of Data Bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing and Prescaler Register BTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing and Prescaler Register FBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant,

independent of the transmitter's error state. The data length code of CAN FD remote frames is transmitted as zero.

In case a M_TTCAN Tx Buffer is configured for CAN FD transmission with DLC > 8, the first 8 bytes are transmitted as configured in the Tx Buffer while the remaining part of the data field is padded with 0xCC. When the M_TTCAN receives a CAN FD frame with DLC > 8, the first 8 bytes of that frame are stored into the matching Rx Buffer or Rx FIFO. The remaining bytes are discarded.

46.4.7.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin m_ttcn_tx the protocol controller receives the transmitted data from its local CAN transceiver via pin m_ttcn_rx. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. Followign figure describes how the transceiver loop delay is measured.

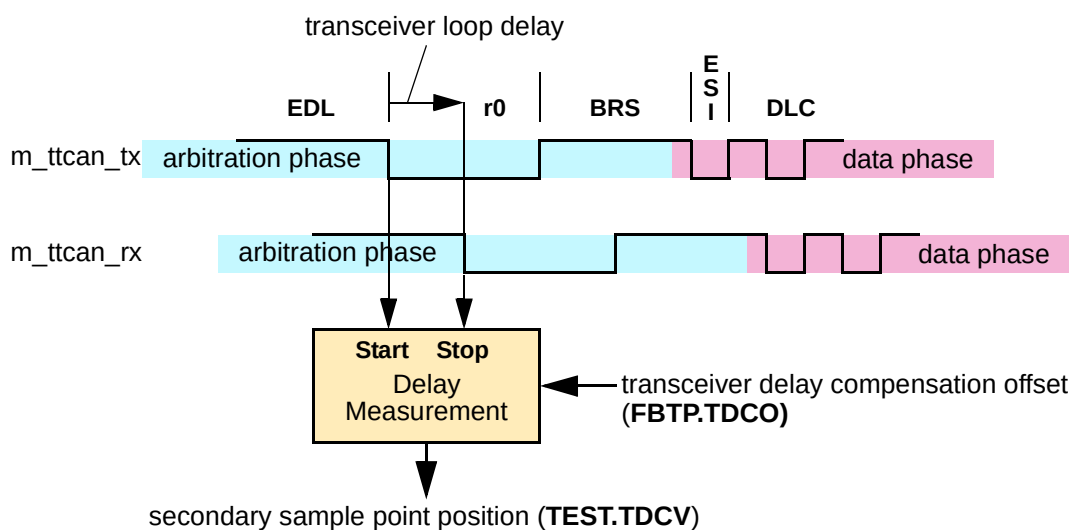


Figure 671. Transceiver delay measurement

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin m_ttcn_tx and the data received at pin m_ttcn_rx, starting with the falling edge of bit EDL. The delay is measured in m_ttcn_cclk periods.

A secondary sample point is calculated by adding a configurable transceiver delay compensation offset (FBTP.TDCO) to the measured transceiver delay. The transceiver delay compensation offset is used to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta tq.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following sample point. During arbitration phase the delay compensation is always disabled.

The maximum delay which can be compensated by the M_TTCAN's delay compensation during the data phase is three bit times.

46.4.7.4.1 Configuration and Status

Compensation for the transceiver loop delay by the M_TTCAN is enabled via FBTP.TDC. The transceiver delay compensation offset is configured via FBTP.TDCO. The actual delay compensation value applied by the M_TTCAN's protocol engine can be read from TEST.TDCV.

46.4.7.5 Bus Monitoring mode

The M_TTCAN is set in Bus Monitoring Mode by programming CCCR.MON to one or when error level S3 (TTOST[EL] = "11") is entered. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_TTCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the M_TTCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_TTCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. [Figure 672](#) shows the connection of signals M_TTCAN transmit output and M_TTCAN transmit output to the M_TTCAN in Bus Monitoring Mode.

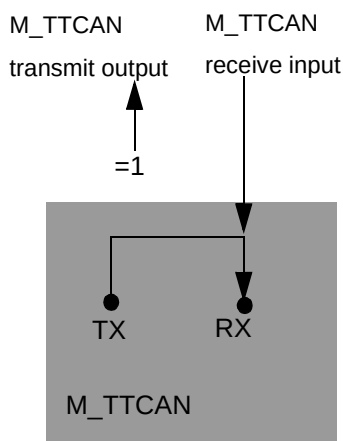


Figure 672. Pin Control in Bus Monitoring Mode

46.4.7.6 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_TTCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic

retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via `CCCR[DAR]`.

46.4.7.7 Frame transmission in DAR mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit `TXBRP.TRPx` is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
 - Corresponding Tx Buffer Transmission Occurred bit `TXBTO[TOx]` set
 - Corresponding Tx Buffer Cancellation Finished bit `TXBCF[CFx]` not set
- Successful transmission in spite of cancellation:
 - Corresponding Tx Buffer Transmission Occurred bit `TXBTO[TOx]` set
 - Corresponding Tx Buffer Cancellation Finished bit `TXBCF[CFx]` set
- Arbitration lost or frame transmission disturbed:
 - Corresponding Tx Buffer Transmission Occurred bit `TXBTO[TOx]` not set
 - Corresponding Tx Buffer Cancellation Finished bit `TXBCF[CFx]` set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type `ET = "10"` (transmission in spite of cancellation).

46.4.7.8 Power down (Sleep mode)

The `M_TTCAN` can be set into power down mode controlled by clock stop request input signal or via CC Control Register `CCCR[CSR]`. As long as the clock stop request signal is active, bit `CCCR[CSR]` is read as one.

When all pending transmission requests have completed, the `M_TTCAN` waits until bus idle state is detected. Then the `M_TTCAN` sets then `CCCR[INIT]` to one to prevent any further CAN transfers. Now the `M_TTCAN` acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and `CCCR[CSA]` to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to `CCCR[INIT]` will have no effect. Now the module clock inputs `M_TTCAN` host clock and `M_TTCAN` clock may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting clock stop request signal respectively CC Control Register flag `CCCR.CSR`. The `M_TTCAN` will acknowledge this by resetting output clock stop acknowledge signal and resetting `CCCR[CSA]`. Afterwards, the application can restart CAN communication by resetting bit `CCCR[INIT]`.

46.4.7.9 Test modes

To enable write access to register `TEST` (see [Section 46.4.5.2.4, Test Register \(TEST\)](#)), bit `CCCR.TEST` has to be set to one. This allows the configuration of the test modes and test functions. Four output functions are available for the `M_TTCAN` transmit pin by programming `TEST.TX`. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the `M_TTCAN`'s bit timing and it can drive constant dominant or recessive values. The actual value at `M_TTCAN` receive pin can be read from `TEST.RX`. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST.TX until the new configuration is visible at output pin. This applies also when reading input pin via TEST.RX.

Note: Test modes should be used for production tests or self test only. The software control for M_TTCAN transmit pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

46.4.7.9.1 Watchdog mode

The application watchdog is served by reading register TTOST. When the application watchdog is not served in time, bit TTOST.AWE is set, all M_TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode.

The TT Application Watchdog can be disabled by programming the Application Watchdog Limit TTOCF[AWL] to 0x00. The TT Application Watchdog should not be disabled in a M_TTCAN application program.

46.4.7.9.2 External Loop Back mode

The M_TTCAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_TTCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. [Figure 673](#) shows the connection of transmit and receive signals to the M_TTCAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_TTCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/ remote frame) in Loop Back mode. In this mode the M_TTCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the input pin is disregarded by the M_TTCAN. The transmitted messages can be monitored at the transmit pin.

46.4.7.9.3 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST.LBCK and CCCR.MON to one. This mode can be used for a “Hot Selftest”, meaning the M_TTCAN can be tested without affecting a running CAN system connected to the M_TTCAN Tx and Rx pins. In this mode pin Rx pin is disconnected from the M_TTCAN and Tx pin is held recessive. [Figure 673](#) shows the connection of M_TTCAN TX and _TTCAN Rx to the M_TTCAN in case of Internal Loop Back Mode.

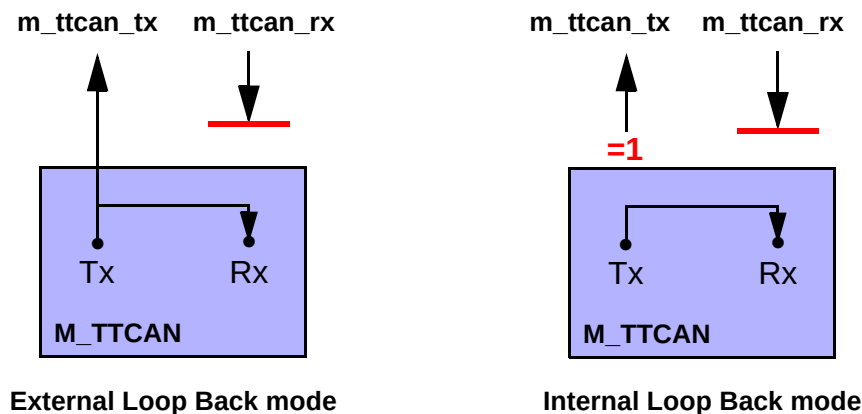


Figure 673. Pin Control in Loop Back modes

46.4.7.10 Timestamp generation

For timestamp generation the M_TTCAN supplies a 16-bit wrap-around counter. A prescaler TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1...16). The counter is readable via TSCV[TCV]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of a Rx Buffer/ Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.

46.4.7.11 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_TTCAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC[TCP] as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC].

The Timeout Counter can only be started while CCCR[INIT] = '0'. It is stopped when CCCR[INIT] = '1', e.g. when the M_TTCAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs/FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: The clock signal for the Timeout Counter is derived from the CAN core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the baud rate

switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

46.4.7.12 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

46.4.7.12.1 Acceptance filtering

The M_TTCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx buffer, Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
 - range filter (from - to)
 - filter for one or two dedicated IDs
 - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

46.4.7.12.2 Range filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames: EFT = "00": The Message ID of received frames is ANDed with the Extended ID AND Mask

(XIDAM) before the range filter is applied EFT = "11": The Extended ID AND Mask (XIDAM) is not used for range filtering

46.4.7.12.3 Filter for dedicated IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID=SF2ID resp.EF1ID=EF2ID.

46.4.7.12.4 Classic bit mask filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID /EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

46.4.7.12.5 Standard message ID filtering

[Figure 674](#) below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in [Section 46.4.6.4, Standard message ID Filter element](#).

Controlled by the Global Filter Configuration (GFC) and the Standard ID Filter Configuration (SIDFC) Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements

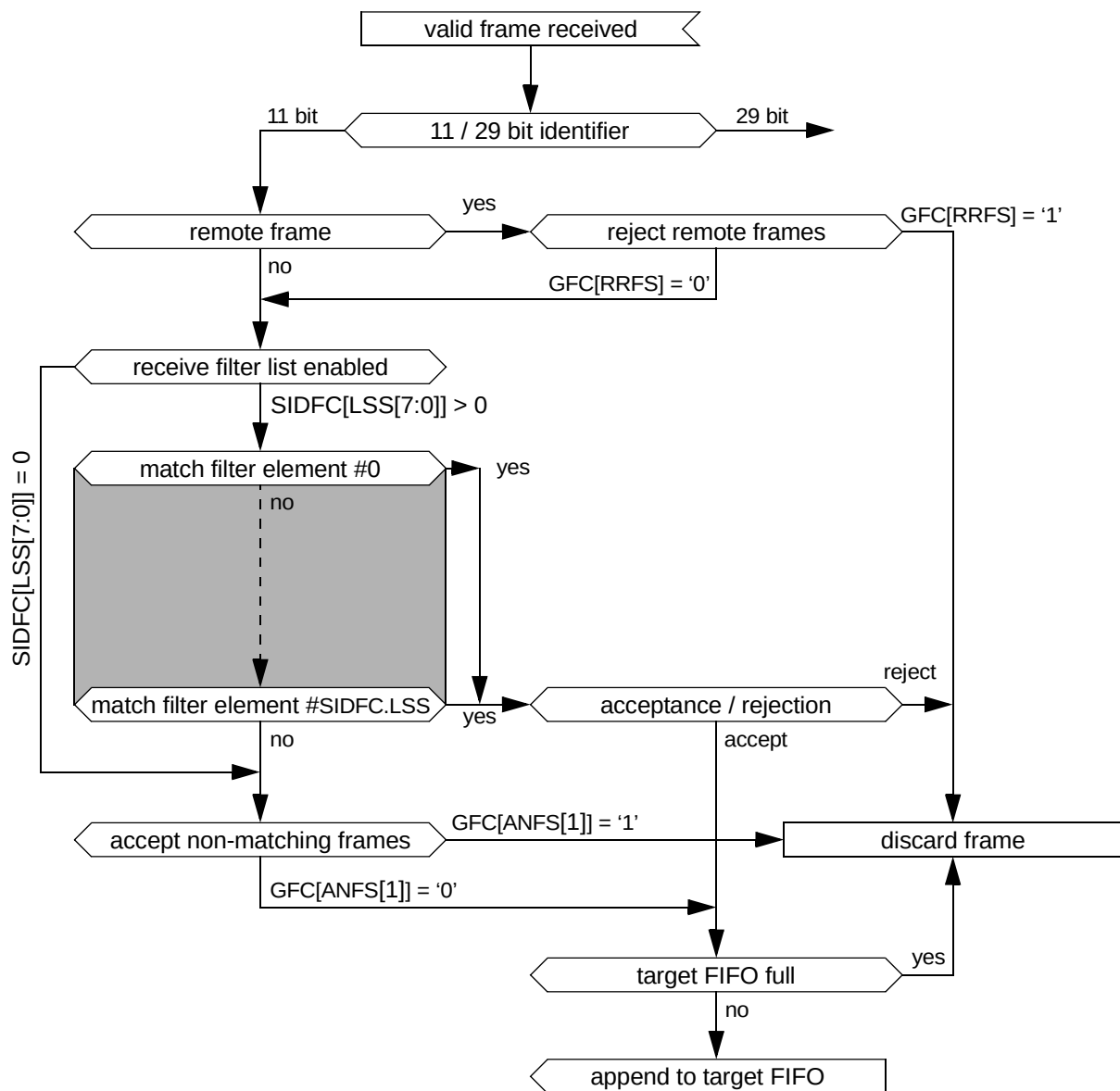


Figure 674. Standard Message ID Filter Path

46.4.7.12.6 Extended message ID filtering

Figure 675 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in [Section 46.4.6.5, Extended message ID filter element](#).

Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements which is controlled by the Global Filter Configuration (GFC) and the Extended ID Filter Configuration (XIDFC) Message ID. The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

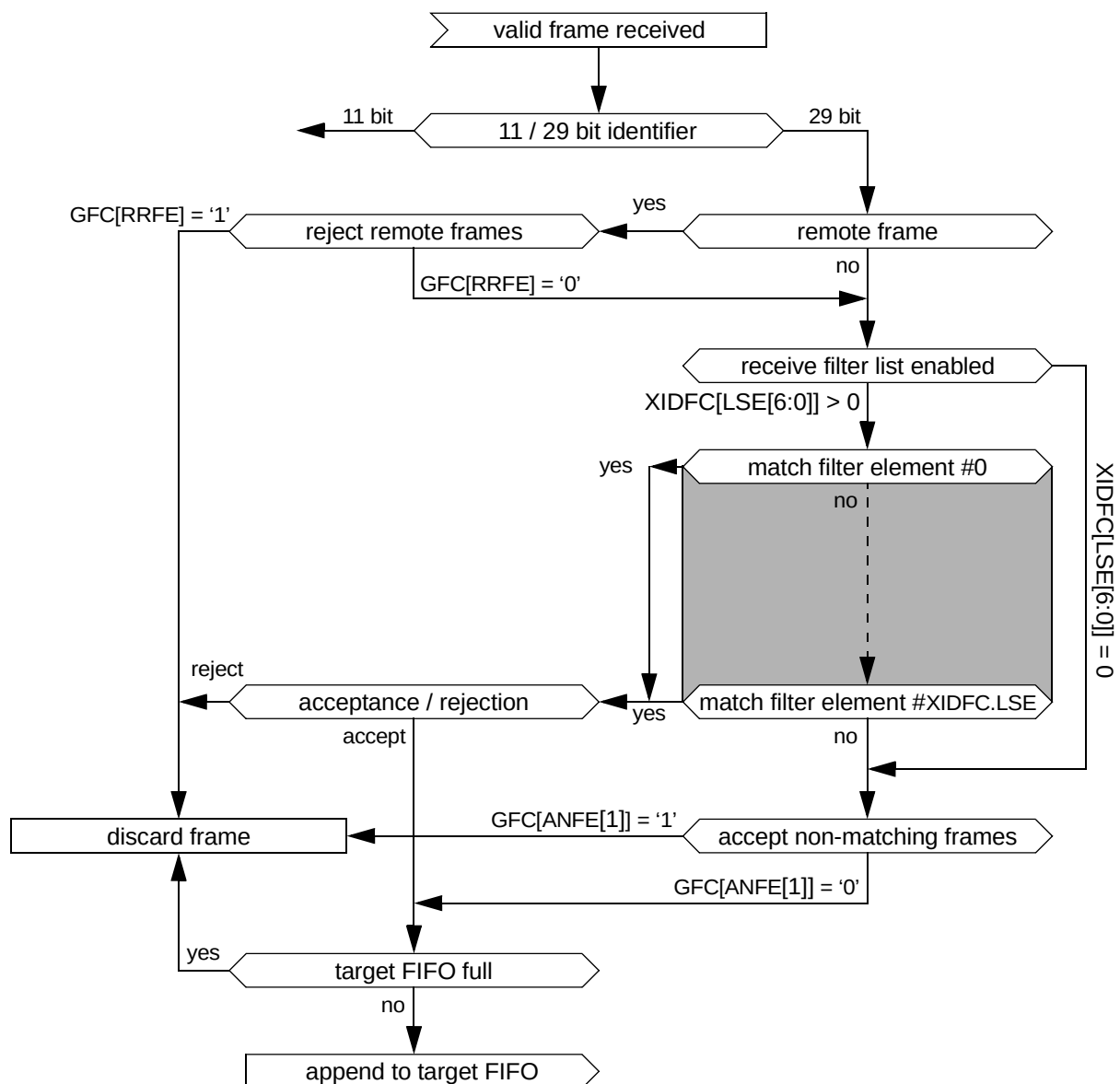


Figure 675. Extended Message ID Filter Path

46.4.7.13 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see [Section 46.4.7.12.1, Acceptance filtering](#). The Rx buffer and FIFO element is described in [Section 46.4.7.13, Rx FIFOs](#).

When an Rx FIFO full condition is signaled by IR[RxFnF]], no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO

Get Index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and interrupt flag IR[RFnL] is set.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC[FnWM], interrupt flag IR[RFnW] is set.

When reading from an Rx FIFO, four times the Rx FIFO Get Index RXFnS[FnGI] has to be added to the corresponding Rx FIFO start address RXFnC[FnSA].

46.4.7.14 Dedicated Rx Buffers

The M_TTCAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = "111" and SFID2 / EFID2[10:9] = "00" has to be configured (see [Section 46.4.6.4, "Standard message ID Filter element"](#) and [Section 46.4.6.5, "Extended message ID filter element"](#)).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 750. Example Filter Configuration for Rx Buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

46.4.7.14.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

46.4.7.15 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in [Section 46.3.6.2, Tx buffer element](#). The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

46.4.7.16 Dedicated Tx buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first. If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of a Dedicated Tx Buffer in the Message RAM is calculated by adding four times the transmit buffer index (0...31) to the Tx Buffer Start Address TXBC[TBSA].

46.4.7.17 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_TTCAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements. New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF]= '1') is signaled.

In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index. When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding four times the Put Index TXFQS[TFQPI] (0...31) to the Tx Buffer Start Address TXBC[TBSA].

46.4.7.18 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first. New messages have to be written to the Tx Buffer referenced by the Put Index

TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF] = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding four times the Tx Queue Put Index TXFQS[TFQPI] (0...31) to the Tx Buffer Start Address TXBC[TBSA].

46.4.7.19 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case, TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

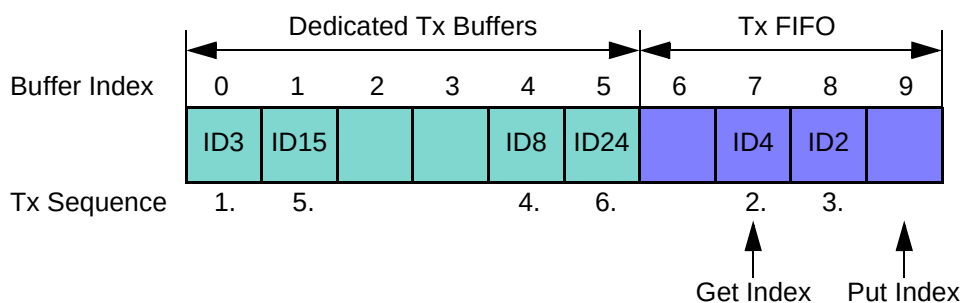


Figure 676. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS[TFGI])
- Buffer with lowest Message ID gets highest priority and is transmitted next

46.4.7.20 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by

TXBC[NDTB]. The number of Tx Queue Buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only dedicated Tx Buffers are used.

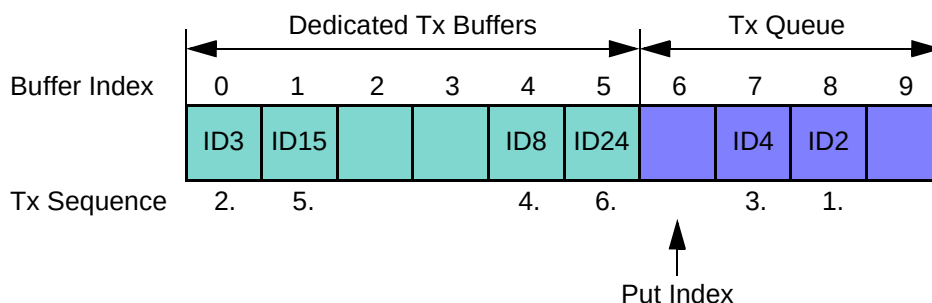


Figure 677. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

46.4.7.21 Transmit cancellation

The M_TTCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation. Successful cancellation is signaled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

46.4.7.22 Tx Event handling

To support Tx event handling the M_TTCAN has implemented a Tx Event FIFO. After the M_TTCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element. The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in [Section 46.3.6.3, Tx event FIFO element](#). When a Tx Event FIFO full condition is signaled by IR[TEFF], no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR[TEFL] is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR[TEFW] is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

46.4.8 FIFO acknowledge handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see [Section 46.4.5.2.27, Rx FIFO 0 Acknowledge Register \(RXF0A\)](#), [Section 46.4.5.2.31, Rx FIFO 1 Acknowledge Register \(RXF1A\)](#), and [Section 46.4.5.2.41, Tx Event FIFO Configuration Register \(TXEFC\)](#)). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index. When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_TTCAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

46.4.9 M_TTCAN operations

46.4.9.1 Reference message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the Time Master (sender of the reference message). For Level 1 the data length must be at least one; for Level 0, 2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to eight potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register TTRMC. The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted **Master_Ref_Mark** is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (**Next_is_Gap** bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters. The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

46.4.9.2 Level 1

Level 1 operation is configured via TTOCF[OM] = “01” and TTOCF[GEN]. External clock synchronization is not available in Level 1. The information related to the reference message is stored in the first data byte as shown below. Cycle_Count is optional.

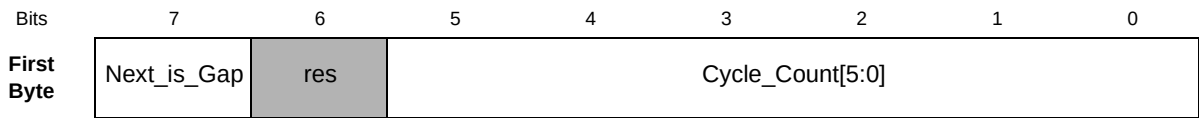


Figure 678. First byte of Level 1 reference message

46.4.9.3 Level 2

Level 2 operation is configured via TTOCF[OM] = “10” and TTOCF[GEN]. The information related to the reference message is stored in the first four data bytes as shown below. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

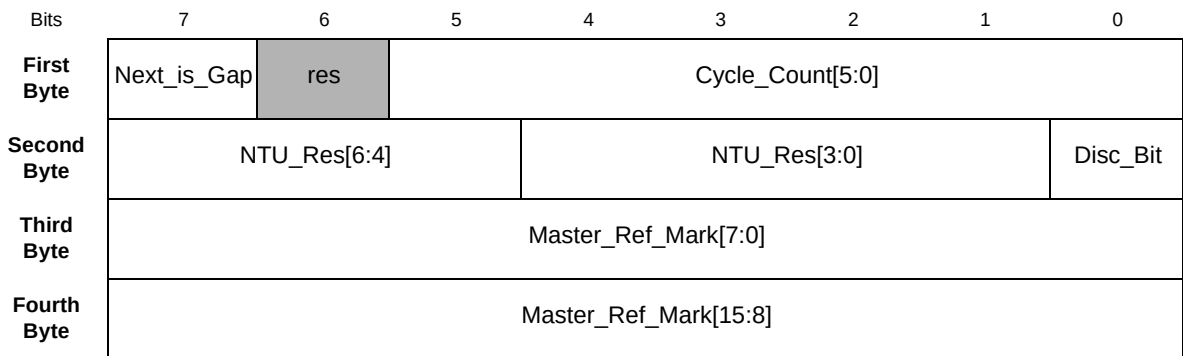


Figure 679. First four bytes of Level 2 reference message

46.4.9.4 Level 0

Level 0 operation is configured via TTOCF[OM] = “11”. External event-synchronized time-triggered operation is not available in Level 0. The information related to the reference message is stored in the first four data bytes as shown in the table below. In Level 0 Next_is_Gap is always zero. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

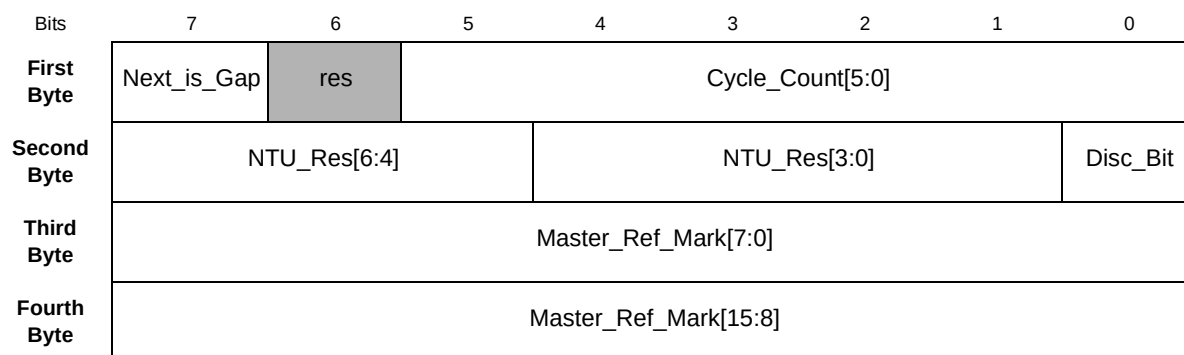


Figure 680. First four bytes of Level 0 reference message

46.4.10 M_TTCAN configuration

46.4.10.1 M_TTCAN timing

The Network Time Unit NTU is the unit in which all times are measured. The NTU is a constant of the whole network and is defined a priority by the network system designer. In M_TTCAN Level 1 the NTU is the nominal CAN bit time. In M_TTCAN Level 0 and Level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable. In M_TTCAN Level 0 and Level 2 the length of the NTU is defined by the Time Unit Ratio TUR. The TUR is in general a non-integer number, given by the formula $TUR = \frac{TURNA[NAV]}{TURCF[DC]}$. The length of the NTU is given by the formula $NTU = \text{CAN Clock Period} \times TUR$. The TUR Numerator Configuration NC is an 18-bit number, $TURCF[NCL[15:0]]$ can be programmed in the range 0x0000–0xFFFF. $TURCF[NCH[17:16]]$ is hard wired to 0b01. When the number 0xn timer is written to $TURCF[NCL[15:0]]$, $TURNA[NAV]$ starts with the value $0x10000 + 0x0nnnn = 0x1nnnn$. The TUR Denominator Configuration $TURCF[DC]$ is a 14-bit number. $TURCF[DC]$ may be programmed in the range 0x0001 - 0x3FFF, 0x0000 is an illegal value.

In Level 1, NC must be $\geq 4 \times TURCF[DC]$. In Level 0, 2 NC must be $\geq 8 \times TURCF[DC]$ to allow the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets $TURCF[DC]$ to 0x1000 and $TURCF[NCL]$ to 0x10000, resulting in an NTU consisting of 16 CAN clock periods. Local time and application watchdog are not started before either the $CCCR[INIT]$ is reset, or $TURCF[ELT]$ is set. $TURCF[ELT]$ may not be set before the NTU is configured. Setting $TURCF[ELT]$ to e1f also locks the write access to register $TURCF$.

At startup $TURNA[NAV]$ is updated from NC ($= TURCF[NCL] + 0x10000$) when $TURCF[ELT]$ is set. In M_TTCAN Level 1 there is no drift compensation. $TURNA.NAV$ does not change during operation, it always equals NC.

In M_TTCAN Level 0 and Level 2 there are two possibilities for $TURNA[NAV]$ to change. When operating as time slave or backup time master, and when $TTOCF[ECC]$ is set, $TURNA[NAV]$ is updated automatically to the value calculated from the monitored global time speed, as long as the M_TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization it returns to NC. When operating as the actual time master,

and when TTOCF[EECS] is set, the Host may update TURCF[NCL]. When the Host sets TTOCN[ECS], TURNA[NAV] will be updated from the new value of NC at the next reference message. The status flag TTOST[WECS] as is set when TTOCN[ECS] is set and is cleared when TURNA[NAV] is updated. TURCF[NCL] is write locked while TTOST[WECS] is set.

In M_TTCAN Level 0 and Level 2 the clock calibration process adapts TURNA[NAV] in the range of the Synchronization Deviation Limit SDL of $NC \pm 2^{(TTOCF[LDSDL]+5)}$. TURCF[NCL] should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of TURNA[NAV].

The synchronization deviation SD is the difference between NC and TURNA[NAV] ($SD = |NC - TURNA[NAV]|$). It is limited by the Synchronization Deviation Limit SDL, which is configured by its dual logarithm TTOCF[LDSDL] ($SDL = 2^{(TTOCF[LDSDL]+5)}$) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new Basic Cycle. When the calculated TURNA[NAV] deviates by more than SDL from NC, or if the Disc_Bit in the Reference Message is set, the drift compensation is suspended and TTIR[GTE] is set and TTOSC[QCS] is reset, or in case of the Disc_Bit = '1', TTIR[GTD] is set.

TUR	8	10	24	50	510	125000	32.5	100/12	529/17
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
TURCF.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

Figure 681. TUR configuration examples

TTOCN[ECS] schedules NC for activation by the next reference message. TTOCN[SGT] schedules TTGTP[TP] for activation by the next reference message. Setting of TTOCN[ECS] and TTOCN[SGT] requires TTOCF[EECS] to be set (external clock synchronization enabled) while the M_TTCAN is actual time master. The M_TTCAN module provides an application watchdog to verify the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The Application Watchdog Limit TTOCF[AWL] specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The Application Watchdog is served by reading register TTOST. TTOST[AWE] indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag TTIR[AW] is set. For software development, the application watchdog may be disabled by programming TTOCF[AWL] to 0x00 (see also Section 46.4.7.9.1).

46.4.10.2 Timing of interface signals

The timing events which cause a pulse at output M_TTCAN trigger time mark interrupt pulse and M_TTCAN register time mark interrupt pulse are generated in the CAN clock domain. There is a clock domain crossing delay to be considered before the same event is visible in the Host clock domain (TTIR[TTMI] resp. TTIR[RTMI] set). The signals can be connected e.g. to the timing input(s) of another M_TTCAN node, in order to automatically synchronize two M_TTCAN networks. Output M_TTCAN start of cycle gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the Host clock domain.

46.4.11 Message scheduling

TTOCF[TM] controls whether the M_TTCAN operates as a potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message's identifier TTRMC[RID] define the master priority, 0 giving the highest and 7 giving the lowest priority. There may not be two nodes in the network using the same master priority. TTRMC[RID] is used for recognition of reference messages. TTRMC[RMPS] is not relevant for time slaves. The Initial Reference Trigger Offset TTOCF[IRTO] is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for TTOCF[IRTO] is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

TTOCF[OM] decides whether the node operates in M_TTCAN Level 0, Level 1, or Level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on Level 1 in a Level 2 network, but not vice versa. TTOCF[EECS] enables the external clock synchronisation, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in Level 0, 2 only) the global clock phase to an external reference.

The configuration of the M_TTCAN operation mode via TTOCF[OM] is the last step in the setup. With TTOCF[OM] = "00" (event-driven CAN communication), the M_TTCAN operates according to ISO 11898-1, without time triggers. With TTOCF[OM] = "01" (Level 1), the M_TTCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the **Next is Gap** bit in the reference message is ignored. With TTOCF[OM] = "10" (Level 2), the M_TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With TTOCF[OM] = "11" (Level 0), the M_TTCAN operates as event-driven CAN but maintains a calibrated global time base as in Level 2.

TTOCF[EECS] enables the external clock synchronisation, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in Level 0,2 only) the global clock phase to an external reference.

TTMLM[ENTT] in the TT Matrix Limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger's repeat factors have to be taken into account. An inaccurate configuration of TTMLM[ENTT] will result in either a TxCount Underflow (TTIR[TXU] = '1' and TTOST[EL] = "01", severity 1) or in a Tx Count Overflow (TTIR[TXO] = '1' and TTOST[EL] = "10", severity 2).

Note: In case the first Reference Message seen by a node does not have Cycle_Count zero, this node may finish its first matrix cycle with its Tx count resulting in a Tx Count Underflow condition. As long as a node is in state Synchronizing its Tx_Triggers will not lead to transmissions.

TTMLM[CCM] specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of 8 basic cycles TTMLM[CCM] would be 7. TTMLM[CCM] is ignored by time slaves, a receiver of a

reference message considers the received cycle count as the valid cycle count for the actual basic cycle. TTMLM[TXEW] specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If a transmission of a message cannot be started inside the Tx enable window because of for example, a slight overlap from the previous time window's message, the transmission cannot be started in that time window at all. TTMLM[TXEW] has to be chosen with respect to the network's synchronisation quality and with respect to the relation between the length of the time windows and the length of the messages.

46.4.11.1 Trigger memory

The trigger memory is part of the external Message RAM to which the M_TTCAN is connected via its Generic Master Interface (see [Figure 585](#)). It stores up to 64 trigger elements. A trigger memory element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, Filter Type FTYPE, Message Number MNR, Message Status Count MSC, Time Mark Event Internal TMIN, Time Mark Event External TMEX, and Asynchronous Serial Communication ASC (see [Section 46.4.6.6, Trigger memory element](#)).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List. When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a transmit trigger, the Tx Handler starts to read the message from the Message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx Handler has to be able to prepare the transmission before the transmit trigger's time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mark is reached. If the reference message is n NTU long, a trigger with a time mark $< n$ will never become active and will be treated as a configuration error. Starting point of the cycle time is the sample point of the reference message's start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger's time mark. The M_TTCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialisation) be seen at a value $< n$, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger's time mark+1 NTU+1 CAN bit time.

The trigger list will be different for all nodes in the M_TTCAN network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

46.4.11.1.1 Trigger types

Tx_Ref_Trigger (**TYPE** = "0000") and Tx_Ref_Trigger_Gap (**TYPE** = "0001") cause the transmission of a reference message by a time master. A configuration error (**TTOST[EL]** = "11", severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronised time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the M_TTCAN

synchronisation state is In_Gap (**TTOST[SYS]** = "10"). Tx_Trigger_Single (**TYPE** = "0010"), Tx_Trigger_Continuous (**TYPE** = "0011"), Tx_Trigger_Arbitration (**TYPE** = "0100"), and Tx_Trigger_Merged (**TYPE** = "0101") cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit is reset. Tx_Trigger_Continuous starts a transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A Configuration Error (**TTOST[EL]** = "11", severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx_Triggers (**TTMLM[ENTT]**) is calculated.

Watch_Trigger (**TYPE** = "0110") and Watch_Trigger_Gap (**TYPE** = "0111") check for missing reference messages. They are used by both time masters and time slaves.

Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the M_TTCAN synchronisation state is In_Gap (**TTOST[SYS]** = "10"). Rx_Trigger (**TYPE** = "1000") is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger shall be placed after the end of that message's transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by **MNR**. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements which are referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger's time mark is reached.

Time_Base_Trigger (**TYPE** = "1001") are used to generate internal/external events depending on the configuration of **ASC**, **TMIN**, and **TMEX**. End_of_List (**TYPE** = "1010...1111") is an illegal trigger type, a configuration error (**TTOST[EL]** = "11", severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger.

46.4.11.1.2 Restrictions for the node's trigger list

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark. Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged or Tx_Ref_Trigger. Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when **TTOST[SYS]** = "10") will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time. All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when **TTOST[SYS]** = "10") must be set to trigger type End_of_List. A

typical trigger list for a potential time master will begin with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave will be the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger's time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the Reference Trigger Offset **TTOCF[IRTO]**. In case error level 2 is reached (**TTOST[EL]** = "10"), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen, that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- Host clock frequency
- Speed and waiting time for Trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

46.4.11.1.3 Example for trigger handling

The example below shows how the trigger list is derived from a node's system matrix. Assumed node A is first time master and has knowledge of the section of the system matrix shown below.

Table 751. System Matrix Node A

Cycle Count	Time Mark1	Time Mark2	Time Mark3	Time Mark4	Time Mark5	Time Mark6	Time Mark7
0	Tx7					TxRef	Error
1	Rx3		Tx2, Tx4			TxRef	Error
2						TxRef	Error
3	Tx7		Rx5			TxRef	Error
4	Tx7			Rx6		TxRef	Error

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the number of basic cycles in the system matrix is 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by

TTMLM.CCM. The Cycle Code CC is composed of repeat factor (= value of most significant '1') and the number of the first basic cycle in the system matrix (= bit field after most significant '1'). Example: with a cycle code of 0b0010011 (repeat factor: 16, first basic cycle: 3) and a maximum cycle count of TTMLM.CCM = "0x3F" matches occur at cycle counts 3, 19, 35, 51.

A trigger element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, and Message Number MNR. For transmission MNR references the Tx Buffer number (0..31). For reception MNR references the number of the filter element (0..127) that matched during acceptance filtering. Depending on the configuration of the Filter Type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition a trigger element can be configured for Asynchronous Serial Communication ASC, generation of Time Mark Event Internal TMIN, and Time Mark Event External TMEX. The Message Status Count MSC holds the counter value (0..7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 752. Trigger List Node A

Trigger	Time Mark TM[15:0]	Cycle Code CC [6:0]	Trigger Type TYPE [3:0]	Mess No. MNR [6:0]
0	Mark 1	0b0000100	Tx_Trigger_Single	7
1	Mark 1	0b1000000	Rx_Trigger	3
2	Mark 1	0b1000011	Tx_Trigger_Single	7
3	Mark 3	0b1000001	Tx_Trigger_Merged	2
4	Mark 3	0b1000011	Rx_Trigger	5
5	Mark 4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark 4	0b1000100	Rx_Trigger	6
7	Mark 6	n.a.	Tx_Ref_Trigger	0 (Ref)
8	Mark 7	n.a.	Watch_Trigger	n.a.
9	n.a.	n.a.	End_of_List	n.a.

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored. The FSE starts the basic cycle with scanning the trigger list starting from zero until a trigger with time mark > cycle time and with its Cycle Code **CC** matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered. When the cycle time reached the Time Mark **TM**, the action defined by Trigger Type **TYPE** and Message Number **MNR** is started. There is an error in the configuration when End_of_List is reached.

At Mark6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the Watch Trigger at Mark7 is reached, the node was not able to transmit the reference message; error treatment is started.

46.4.11.1.4 Detection of configuration errors

A configuration error is signaled via **TTOST[EL]** = "11" (severity 3) when: The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time. The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger. The FSE of a node with **TTOCF[TM]** = '0' (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.

Any time mark placed inside the Tx enable window (defined by **TTMLM[TXEW]**) of a Tx_Trigger with a matching cycle code. A time mark is placed near the time mark of a Tx_Ref_Trigger and the Reference Trigger Offset **TTOST[RTO]** causes a reversal of their sequential order measured in cycle time.

46.4.11.2 M_TTCAN schedule initialization

The synchronisation to the M_TTCAN's message schedule starts when **CCCR[INIT]** is reset. The M_TTCAN can operate strictly time-triggered (**TTOCF[GEN]** = '0') or external event-synchronized time-triggered (**TTOCF[GEN]** = '1'). All nodes start with cycle time zero at the beginning of their trigger list with **TTOST[SYS]** = "00" (out of synchronisation), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronised time-triggered operation mode will ignore Tx_Ref_Trigger and Watch_Trigger and will use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

46.4.11.2.1 Time slaves

After configuration, a time slave will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag **TTIR[IWT]** is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

Note: Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter which counts up to 0xFFFF = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it will assume a fatal error (**TTOST[EL]** = "11", severity 3), set interrupt flag **TTIR[WT]**, switch off its CAN bus output, and enter the bus monitoring mode (**CCCR[MON]** set to '1'). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

Note: To leave the fatal error state, the Host has to set CCCR[INIT] = '1'. After reset of CCCR[INIT], the node restarts M_TTCAN communication.

When no error is encountered during synchronisation, the first reference message sets **TTOST[SYS]** = "01" (Synchronizing), the second sets the M_TTCAN synchronization state (depending on its Next_is_Gap bit) to **TTOST[SYS]** = "11" (In_Schedule) or **TTOST[SYS]** = "10" (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

46.4.11.2.2 Potential time masters

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch_Trigger and

Watch_Trigger_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag **TTIR[IWT]** is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting **TTIR[IWT]** will re-enable the transmission of reference messages until next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE will not be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it will assume a fatal error (**TTOST[EL]** = "11", severity 3), set interrupt flag **TTIR[WT]**, switch off its CAN bus output, and enter the bus monitoring mode (**CCCR[MON]** set to '1'). In bus monitoring mode, it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets **TTOST[SYS]** = "01" (synchronizing), the second sets the M_TTCAN synchronization state (depending on its **Next_is_Gap** bit) to **TTOST[SYS]** = "11" (In_Schedule) or **TTOST[SYS]** = "10" (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (**TTOST[MS]** = "11") when it was the transmitter of the last reference message, else it is backup time master (**TTOST[MS]** = "10").

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

46.4.12 M_TTCAN gap control

All functions related to Gap control apply only when the M_TTCAN is operated in external event synchronized time-triggered mode (**TTOCF[GEN]** = '1'). In this operation mode the M_TTCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event- synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit **Next_is_Gap** = '1' e.g. Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes **TTOCN[NIG]** = '1'. The **Next_is_Gap** bit will be transmitted as '1' with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input pin by writing **TTOCN[GCS]** = '1'. When a reference message is started and **TTOCN[GCS]** is set, a HIGH level at event trigger pin will set **Next_is_Gap** = '1'.

As soon as that reference message is completed, the **TTOST[WFE]** bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, **TTOST[GS]** will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit **TTOST[GS]** will remain at '0'.

When a potential time master is in synchronization state `In_Gap` (`TTOST[SYS] = "10"`), it has four options to intentionally finish a Gap:

Under software control by writing `TTOCN[FGP] = '1'`.

Under hardware control (`TTOCN[GCS] = '1'`) an edge from HIGH to LOW at the event-trigger input pin sets `TTOCN[FGP]` and restarts the schedule.

The third option is a time-triggered restart. When `TTOCN[TMG] = '1'`, the next register time mark interrupt (`TTIR[RTMI] = '1'`) will set `TTOCN[FGP]` and start the reference message.

Finally any potential time master will finish a Gap when it reaches its `Tx_Ref_Trigger_Gap`, assuming that the event to synchronize on did not occur in time.

Neither of these options can cause a basic cycle to be interrupted with a reference message. Setting of `TTOCN[FGP]` after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the message schedule. When `TTOCN[FGP]` is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the `Tx_Ref_Trigger` – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit `Next_is_Gap` = '1' in the reference message will be ignored, as well as the event-trigger input pin and the bits `TTOCN[NIG]`, `TTOCN[FGP]`, and `TTOCN[TMG]`.

46.4.13 Stop watch

The stop watch function enables capturing of M_TTCAN internal time values (local time, cycle time, or global time) triggered by an external event. To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via `TTOCN[SWS]`. When `TTOCN[SWS]` is $\neq$ '00' and TT Interrupt Register flag `TTIR[SWE]` is '0', the actual value of the time selected by `TTOCN[SWS]` will be copied into `TTCPT[SWV]` on the next rising/falling edge (as configured via `TTOCN[SWP]`) on stop watch trigger pin. This will set interrupt flag `TTIR[SWE]`. After the application program has read `TTCPT[SWV]`, it may enable the next stop watch event by resetting `TTIR[SWE]` to '0'.

46.4.14 Local time, cycle time, global time, and external clock synchronization

There are two possible levels in time-triggered CAN: Level 1 and Level 2. Level 1 only provides time-triggered operation using cycle time. Level 2 additionally provides increased synchronisation quality, global time and external clock synchronisation. In both levels, all timing features are based on a local time base - the local time. The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented 8 times each NTU. If the length of the NTU is shorter than 8 CAN clock periods (as may be configured in Level 1, or as a result of clock calibration in Level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only 4 times each NTU.

[Figure 682](#) describes the synchronisation of the cycle time and global time, performed in the same manner by all M_TTCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message is frame synchronisation event. This frame synchronisation event occurs at the sample point of each Start of Frame (SoF) bit and causes the local time to be stored as `Sync_Mark`. `Sync_Marks` and `Ref_Marks` are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the Cycle Sync Mark (Cycle Sync Mark = Sync_Mark - Ref_Mark) stored in register **TTCSM**. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (Cycle Time = Local Time - Ref_Mark).

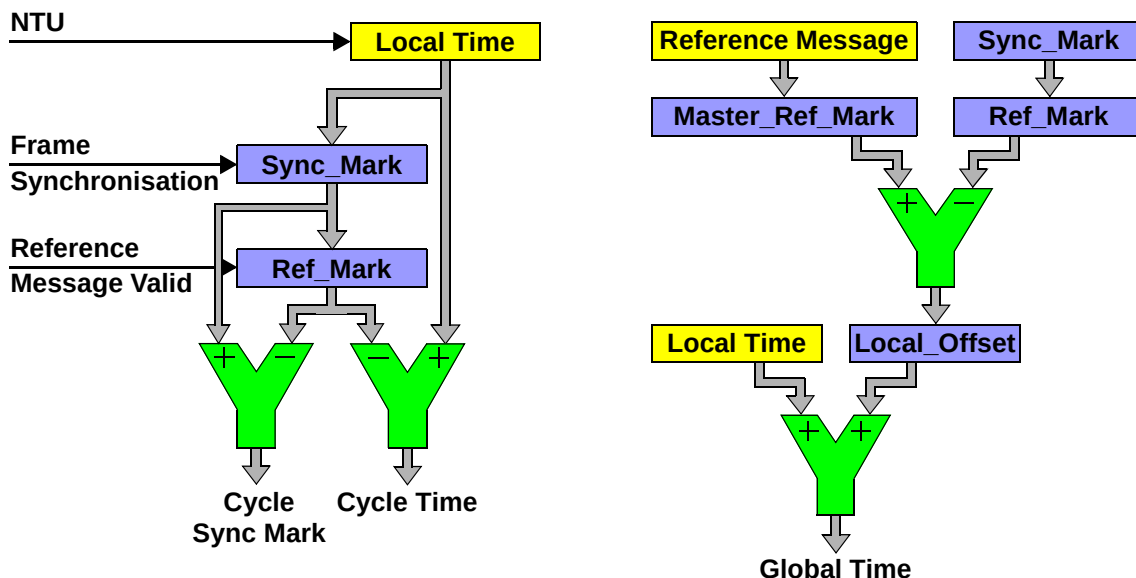


Figure 682. Cycle Time and Global Time Synchronization

The cycle time that can be read from **TTCTC[CT]** is the difference of the node's local time and Ref_Mark, both synchronized into the Host clock domain and truncated to 16 bit. The global time exists for M_TTCAN Level 0 and Level 2 only, in Level 1 it is invalid. The node's view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3,4). The global time that can be read from **TTLGT[GT]** is the sum of the node's local time and its local offset, both synchronized into the Host clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see in figure above). The node's view of the global time is local time + local offset. In a potential time master that has never received another time master's reference message, Local_Offset will be zero. When a node becomes the current time master after first having received other reference messages, Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signaled by **Disc_Bit** in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register **TTLGT** is smoothed by a low-pass filtering to have a continuous monotonic value.

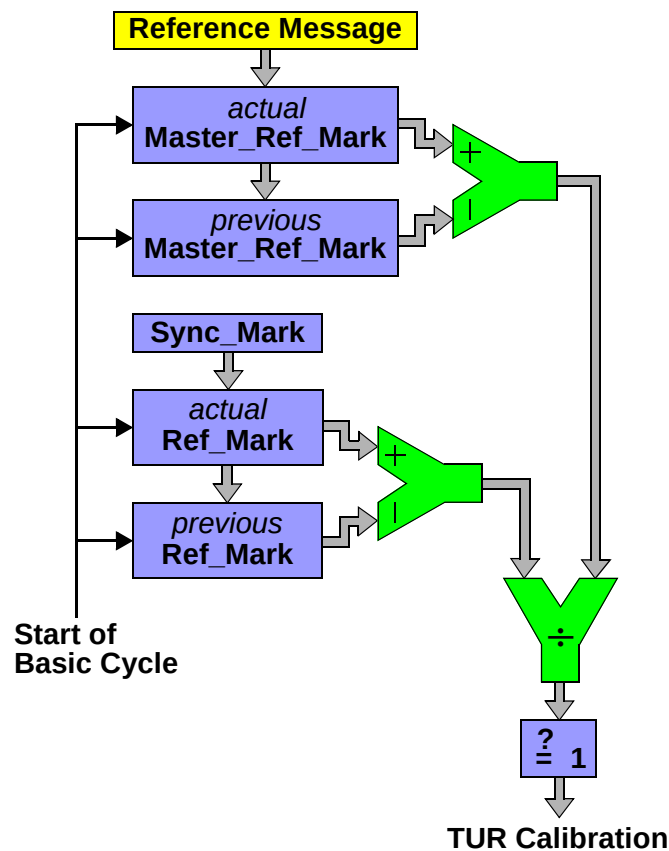


Figure 683. M_TTCAN Level 0 and Level 2 Drift Compensation

Figure 683 describes how in M_TTCAN Level 0,2 each time receiving node compensates the drift between its own local clock and the time master's clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the **Disc_Bit** in the reference message is not set, a new value for **TURNA[NAV]** is calculated. If the Synchronisation Deviation $SD = |NC - TURNA[NAV]| \leq SDL$ (Synchronisation Deviation Limit), the new value for **TURNA[NAV]** takes effect. Else the automatic drift compensation is suspended.

In M_TTCAN Level 0 and Level 2, **TTOST[QCS]** indicates whether the automatic drift compensation is active or suspended. In M_TTCAN Level 1, **TTOST[QCS]** is always '1'.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit **TTOCF[EECS]**.

The stop watch function (see [Section 46.4.13, Stop watch](#)) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated Numerator Configuration Low to **TURCF[NCL]** (**TURCF[DC]** cannot be updated during operation). The new value takes effect by writing **TTOCN[ECS]** to '1'.

The global time phase is adjusted by first writing the phase offset into the TT Global Time Preset register **TTGTP**. The new value takes effect by writing **TTOCN[SGT]** to '1'. The first reference message transmitted after the global time phase adjustment will have the **Disc_Bit** set to '1'.

TTOST[QGTP] shows whether the node's global time is in phase with the time master's global time. **TTOST[QGTP]** is permanently '0' in M_TTCAN Level 1 and when the Synchronisation Deviation Limit is exceeded in M_TTCAN Level 0,2 (**TTOST[QCS]** = '0'). It is temporarily '0' while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a **Disc_Bit** = '1' or when **TTOST[QCS]**='0'.

46.4.15 M_TTCAN error level

The ISO 11898-4 specifies four levels of error severity:

- S0 - No Error
- S1 - Warning- Only notification of application, reaction application-specific.
- S2 Error- Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the Reference Trigger Offset **TTOST[RTO]** set to the maximum value of 127.
- S3 - Severe Error
Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and **CCCR[MON]** is set. The S3 error condition remains active until the application updates the configuration (set **CCCR[CCE]**).
If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by **TTIR[ELC]**. The error level is monitored by **TTOST[EL]**.

The M_TTCAN signals the following error conditions as required by ISO 11898-4:

- Config_Error (S3)
Sets Error Level **TTOST[EL]** to "11" when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.
- Watch_Trigger_Reached (S3)
Sets Error Level **TTOST[EL]** to "11" when a watch trigger was reached because the reference message is missing.
- Application_Watchdog (S3)
Sets Error Level **TTOST[EL]** to "11" when the application failed to serve the application watchdog. The application watchdog is configured via **TTOCF[AWL]**. It is served by reading register TTOST. When the watchdog is not served in time, bit **TTOST[AWE]** and interrupt flag **TTIR[AW]** are set, all M_TTCAN communication is stopped, and the M_TTCAN is set into bus monitoring mode (**CCCR[MON]** set to '1').
- CAN_Bus_Off (S3)
Entering **CAN_Bus_Off** state sets error level **TTOST[EL]** to "11". **CAN_Bus_Off** state is signaled by **PSR[BO]** = '1' and **CCCR[INIT]** = '1'.
- Scheduling_Error_2 (S2)
Sets Error Level **TTOST[EL]** to "10" if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag **TTIR[SE2]** is set. The Error Level **TTOST[EL]** is reset to "00" at

the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.

- **Tx_Overflow (S2)**
Sets Error Level **TTOST[EL]** to “10” when the Tx count is equal or higher than the expected number of Tx_T triggers **TTMLM[ENTT]** and a Tx_Trigger event occurs. In addition interrupt flag **TTIR[TXO]** is set. The Error Level **TTOST[EL]** is reset to “00” when the Tx count is no more than **TTMLM[ENTT]** at the start of a new matrix cycle.
- **Scheduling_Error_1 (S1)**
Sets Error Level **TTOST[EL]** to “01” if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than 2, or if one of the MSCs of an exclusive Rx_Trigger has reached 7. In addition interrupt flag **TTIR[SE1]** is set. If within one matrix cycle none of these conditions is valid, the Error Level **TTOST[EL]** is reset to “00”.
- **Tx_Underflow (S1)**
Sets Error Level **TTOST[EL]** to “01” when the Tx count is less than the expected number of Tx_Triggers **TTMLM[ENTT]** at the start of a new matrix cycle. In addition interrupt flag **TTIR[TXU]** is set. The Error Level **TTOST[EL]** is reset to “00” when the Tx count is at least **TTMLM[ENTT]** at the start of a new matrix cycle.

46.4.16 M_TTCAN message handling

46.4.16.1 Reference message

For potential time masters the identifier of the reference message is configured via **TTRMC[RID]**. No dedicated Tx Buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte (M_TTCAN Level 1) resp. the first four data bytes (M_TTCANM_TTCAN Level 0 and Level 2) will be provided by the FSE. In case the Reference Message Payload Select **TTRMC[RMPS]** is set, the rest of the reference message's payload (Level 1: bytes 2-8, Level 0,2: bytes 5-6) is taken from Tx Buffer 0. In this case the data length **DLC** code from message buffer 0 is used.

Table 753. Number of Data Bytes transmitted with a Reference Messages

TTRMC.RMPS	TXBRP.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4
1	0	4	1	4
1	1	4+MBO	1+MBO	4+MBO

To send additional payload with the reference message in Level 1 a **DLC > 1** has to be configured, for Level 0, 2 a **DLC > 4** is required. In addition the transmission request pending bit **TXBRP[TRP0]** of message buffer 0 must be set (see [Table 753](#)). In case bit **TXBRP[TRP0]** is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the Reference Identifier **TTRMC[RID]** is used.

46.4.16.2 Message reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see [Section 46.3.10, Rx handling](#)).

The Message Status Count MSC is part of the corresponding trigger memory element and has to be initialized to zero during configuration. It is updated while the M_TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message's Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

46.4.16.3 Message transmission

For time-triggered message transmission the M_TTCAN supplies 32 dedicated Tx buffers (see [Section 46.4.7.16, Dedicated Tx buffers](#)). A Tx FIFO or Tx queue is not available when the M_TTCAN is configured for time-triggered operation (**TTOCF[OM]** = "01" or "10"). Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle. The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has completed by reading **TXBTO**
- Update the Tx buffer's configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

Tx_Trigger_Continuous

- Issue a Cancellation Request to reset the Tx Buffer Request Pending bit
- Check whether the cancellation has finished by reading **TXBCF**
- Update Tx buffer's configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

The message's **MSC** stored with the corresponding Tx_Trigger provides information on the success of the transmission. The **MSC** is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The **MSC** is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (M_TTCAN in Error Level S2 or Host has disabled this particular message). The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to

assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking **TXBRP**. If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading **TXBCF** before it starts updating.

The Tx Handler will transfer a message from the Message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be example of a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx Handler can read four words from the Message RAM even at high RAM access load from other modules.

46.4.16.3.1 Transmission in exclusive time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The **MSC** is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx Buffer Request Pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx Buffer Request Pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

46.4.16.3.2 Transmission in arbitrating time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The **MSC** is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

46.4.16.3.3 Transmission in merged arbitrating time windows

The purpose of a merged arbitrating time window is, to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a single node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding Transmission Request Pending flag was reset by a successful Tx cancellation. In single transmit windows, the Tx Handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is reached before the first is transmitted (or cancelled by the Host), the fourth request will be ignored. The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if their is compelling bus traffic. This has to be considered for the configuration of the trigger list.

Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx Buffer needs to be updated.

46.4.17 M_TTCAN interrupt and error handling

The TT Interrupt Register **TTIR** consists off our segments. Each interrupt can be enabled separately by the corresponding bit in the TT Interrupt Enable register **TTIE**. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. The first segment consists of flags **CER**, **AW**, **WT**, and **IWT**. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of **IWT**, these error conditions require a re-configuration of the M_TTCAN module before the communication can be restarted.

The second segment consists of flags **ELC**, **SE1**, **SE2**, **TXO**, **TXU**, and **GTE**. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the M_TTCAN protocol's failure handling and do not require intervention by the application program.

The third segment consists of flags **GTD**, **GTW**, **SWE**, **TTMI**, and **RTMI**. The first two flags are controlled by global time events (Level 0,2 only) that require a reaction by the application program. With a Stop Watch Event triggered by a rising edge on pin internal time values are captured. The Trigger Time Mark Interrupt notifies the application that a specific Time_Base_Trigger is reached. The Register Time Mark Interrupt signals that the time referenced by **TTOCN[TMC]** (Cycle, Local, or Global) equals time mark **TTTMK[TM]**. It can also be used to finish a Gap. The fourth segment consists of flags **SOG**, **CSM**, **SMC**, and **SBC**. These flags provide a means to synchronize the application program to the communication schedule.

46.4.18 Level 0

M_TTCAN Level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in M_TTCAN Level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1. Level 0 operation is configured via **TTOCF[OM] = "11"**. In this mode the M_TTCAN operates in event driven CAN communication, there is no fixed schedule, the configuration of **TTOCF[GEN]** is ignored. External event-synchronized operation is not available in Level 0. A synchronized time base is maintained by transmission of reference messages. In Level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (**TTIR[TTMI]**) is set when the cycle time has reached **TTOCF[IRTO] × 0x200**, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (**TTIR[WT]**) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks. Register time mark interrupts (**TTIR[RTMI]**) are also possible. The reference message is configured as for Level 2 operation. Received reference messages are recognized by the identifier configured in register TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- TTRMC
- **TTOCF** except **EVTP**, **AWL**, **GEN**
- **TTMLM** except **ENTT**, **TXEW**
- TURCF

Level 0 operation is controlled via:

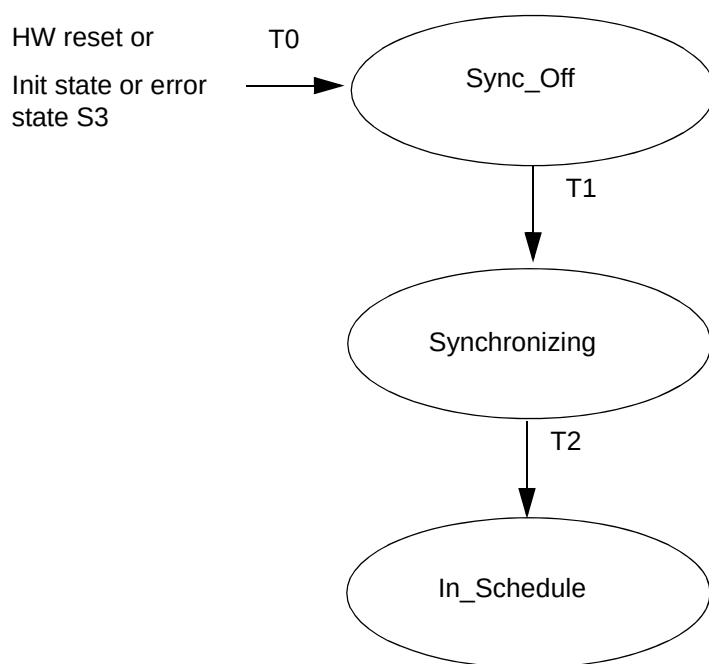
- **TTOCN** except **NIG**, **TMG**, **FGP**, **GCS**, **TTMIE**
- TTGTP
- TTTMK
- **TTIR** excluding bits **CER**, **AW**, **IWT SE2**, **SE1**, **TXO**, **TXU**, **SOG** (no function)
- **TTIR** the following bits have changed function
 - **TTMI** not defined by trigger memory - activated at cycle time **TTOCF[IRTO]** .
0x200
 - **WT** not defined by trigger memory - activated at cycle time 0xFF00

Level 0 operation is signaled via:

- TTOST excluding bits **AWE**, **WFE**, **GSI**, **GFI**, **RTO** (no function)

46.4.18.1 Synchronizing

[Figure 684](#) describes the states and state transitions in M_TTCAN Level 0 operation. Level 0 has no In_Gap state.



T0: transition condition always taking prevalence

T1: Init state left, cycle time is zero

T2: at least two successive reference messages observed (last reference message did not contain a set Disc_Bit or Next_is_Gap bit)

Figure 684. Level 0 schedule synchronization state machine

46.4.18.2 Handling of error levels

During Level 0 operation only the following error conditions may occur:

- Watch_Trigger_Reached (S3), reached cycle time 0xFF00
- CAN_Bus_Off (S3)

Since no S1 and S2 error are possible, the error level can only switch between S0 (No Error) and S3 (Severe Error). In M_TTCAN Level 0 an S3 error is handled differently. When error level S3 is reached, both **TTOST[SYS]** and **TTOST[MS]** are reset, and interrupt flags **TTIR[GTE]** and **TTIR[GTD]** are set.

When error level S3 (**TTOST[EL]** = "11") is entered, bus monitoring mode is, contrary to M_TTCAN Level 1 and Level 2, not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

46.4.18.3 Master slave relation

[Figure 685](#) below describes the master slave relation in M_TTCAN Level 0. In case of an S3 error the M_TTCAN returns to state Master_Off.

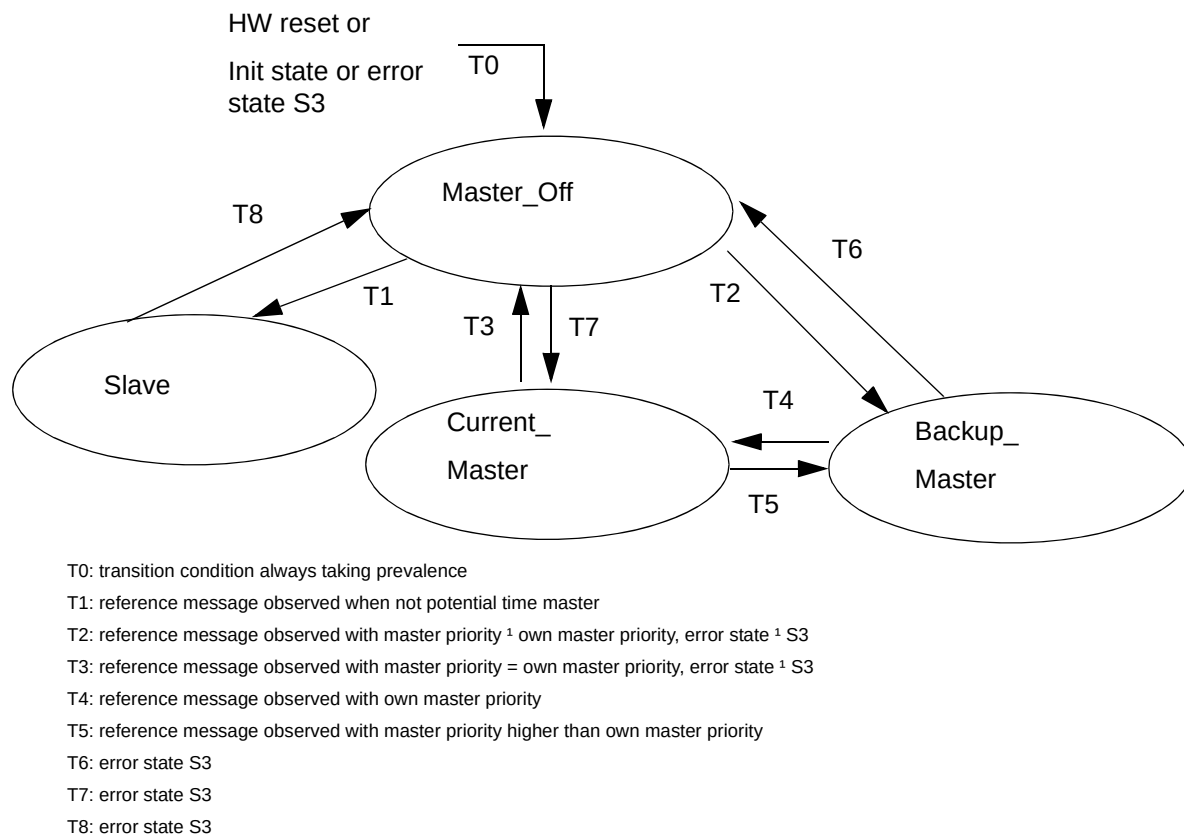


Figure 685. Level 0 master to slave relation

46.4.19 Synchronization to external time schedule

This feature can be used to synchronize the phase of the M_TTCAN's schedule to an external schedule (e.g. that of a second M_TTCAN network or FlexRay network). It is applicable only when the M_TTCAN is current time master (**TTOST[MS]** = "11"). External synchronization is controlled by event trigger input pin. If bit **TTOCN[ESCN]** is set, a rising edge at event trigger pin the M_TTCAN compares its actual cycle time with the target phase value configured by **TTGTP[CTP]**. Before setting **TTOCN[ESCN]** the Host has to adapt the phases of the two time schedules e.g. by using the M_TTCAN gap control (see [Section 46.4.12, M_TTCAN gap control](#)). When the Host sets **TTOCN[ESCN]**, **TTOST[SPL]** is set.

If the difference between the cycle time and the target phase value **TTGTP[CTP]** at the rising edge at event trigger pin is greater than 9 NTU, the phase lock bit **TTOST[SPL]** is reset, and interrupt flag **TTIR[CSM]** is set. **TTOST[SPL]** is also reset (and **TTIR[CSM]** is set), when another node becomes time master. If both **TTOST[SPL]** and **TTOCN[ESCN]** are set, and if the difference between the cycle time and the target phase value **TTGTP[CTP]** at the rising edge at event trigger pin is less or equal 9 NTU, the phase lock bit **TTOST[SPL]** remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

Note: The rising edge detection at event trigger pin is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with TTGTP[CTP]. All further edges until the beginning of the next basic cycle are ignored.

46.5 CAN RAM arbiter

This block acts as a dynamic arbiter between the various CAN nodes and the CPU for granting access to the shared CAN memory. The dynamic arbitration ensures:

- 50% bandwidth for CPU, 50% shared by the CAN nodes when simultaneous access requested from all masters and CPU,
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access.

Note: From the arbiter prospective the CPU bandwidth may be limited to 50%. The CPU may be waiting stated due to this bandwidth limitation when accessing the shared memory.

46.5.1 Features

- Active low asynchronous reset
- 1–CPU + 5–CAN arbiter.
- Combined dynamic arbitration scheme: Bandwidth between CPU and M_CAN interfaces can be redistributed. If active, the CPU interface is granted a 50% portion of the available bandwidth.
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access
- The SRAM access by a single master needs two cycles (address and data) for each read/write command. The arbiter has a pseudo address pre-fetching mechanism. This mechanism acts when multiple masters are accessing the SRAM and it overlaps the address cycle of one master with the data cycle of another master. In case of multiple masters accessing the SRAM, the pseudo address pre-fetching scheme saves multiple clock cycles.
- The peripheral GSI module enable acts as a request from each master. Each master will be made to wait at least one clock cycle before the grant is given. The arbiter ensures that the CPU does not wait for more than 1 clock cycle, hence ensuring a 50% guaranteed bandwidth.
- Time slot is defined as one peripheral clock cycle.

46.5.2 Functional overview using examples

The dynamic arbitration scheme is explained using the following examples.

- Example 1 (active CPU IF, 3 out of 3 CAN nodes are active):
 - CPU obtains every 2nd slot
 - CAN 1/2/3 obtains every 6th time slot
- Example 2 (inactive CPU IF, 4 out of 4 CAN nodes are active):
 - CPU obtains no time slots
 - CAN 1/2/3/4 obtains every 4th time slot
- Example 3 (active CPU IF, 2 out of 4 CAN node are active):
 - CPU obtains every 2nd time slot
 - Active CAN 1/2 obtains every 4th time slot
- Example 4 (inactive CPU IF, 2 out of 4 CAN node are active):
 - CPU obtains no time slots
 - Active CAN 1/2 obtains every 2nd time slot

46.6 SRAM interface and memory organization

The CAN subsystem will interface with an external RAM using this interface. The RAM sizes required for the CAN subsystem are summarized in the following sections.

46.6.1 Shared memory

The following table/s shows maximum possible configuration per Filter, FIFO and Buffer but not all of these items can be configured to maximum size at the same time due to the smaller implemented physical memory as listed in [Table 756](#).

Table 754. M_CAN message memory size

Memory element	Number of elements	32-bit word per element	Total 32-bit words
11-bit Filter	128	1	128
29-bit Filter	64	2	128
Rx FIFO 0	64	4	256
Rx FIFO 1	64	4	256
Rx Buffer	64	4	256
Tx Event FIFO	32	2	64
Tx Buffers	32	4	128
M_CAN message memory size (word)			1216

Please refer [Table 756](#) for the available physical memory space used in the device.

Table 755. M_TTCAN message memory size

Memory element	Number of elements	32-bit word per element	Total 32-bit words
11-bit Filter	128	1	128
29-bit Filter	64	2	128
Rx FIFO 0	64	4	256
Rx FIFO 1	64	4	256
Rx Buffer	64	4	256
Tx Event FIFO	32	2	64
Tx Buffers	32	4	128
Triggers	64	2	128
M_TTCAN message memory size (word)			1344

Please refer [Table 756](#) for the exact physical memory space used in the device.

46.6.2 ECC controller

The RAM embeds ECC logic and code to provide Single Error Correction / Double Error Detection (SECCDED). The module does not guarantee any proper functionality if more than 2 bits are in errors. It supports only 32-bit write access and 8-/16-/32-bit read access. The ECC error is reported to each CAN module as well as available at the CAN subsystem top, so that the same can be forwarded to the Error Management Module.

The module uses Hamming code for single error correction and double error detection logic. It involves transmitting data with multiple ecc_check bits and decoding the associated ecc_check bits when receiving data to detect errors. The SECCDED Hamming code is not able to detect three bit errors. Rather, in the presence of a three bit error, a conventional SECCDED code returns an error code that is indistinguishable from an error code resulting from a single bit error. Hence, this module does not guarantee any proper functionality when more than 2-bits are in error.

The ecc_check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several ecc_check bits show parity errors after decoding the retrieved codeword. The combination of these ecc_check bit errors display the nature of the error. In addition, the position of any single bit error is identified from the ecc_check bits.

The ECC error address is reported at the top of the subsystem using a 16 bit ECC error address signal. This address is valid only when the ECC bit error output is valid.

46.6.2.1 Features

The ECC module supports the following:

- `ecc_error = 00` ' No error is detected in the read data from memory
- `ecc_error = 01` ' This indicates single bit error occurred within the 39-bit codeword. In addition, the error is corrected, and the forwarded data to the master is error free.
- `ecc_error = 10` ' This indicates that a two bit error has occurred within the codeword. In this case, no error correction is possible.
- `ecc_error = 11` ' This indicates that errors beyond the detection capability have occurred within the codeword and no error correction is possible. This is an invalid error type.

46.6.3 Transmit message buffer locking mechanism

Note: Please refer to Device Configuration chapter to see the number of CAN nodes used in the device.

This block will implement a transmit message buffer locking mechanism and trigger memory elements when the M_TTCAN is outside the INIT state using the CAN nodes internal flags output.

The Host controller has to access the Message RAM for updating transmit message buffers (e.g. the data section), the M_CAN controller needs access to the Message RAM to execute acceptance and transmit filtering in time and to prepare the next message for transmission by transferring it to the CAN core.

To avoid message buffer corruption, further host accesses to message buffers which are requested for transmission should be prevented. Otherwise, if the Host updates a message buffer which is simultaneously transferred to the CAN core, inconsistent data may be transmitted on the CAN bus.

This functionality is achieved using the “transmit buffer request pending” (M_CAN) and “configuration change enable” (M_TTCAN) information of the CAN modules. With the information from these signals, the message buffer locker prevents Host accesses from transmitting message buffers which are requested and pending for transmission and trigger memory elements when the M_TTCAN is outside the INIT state.

46.6.3.1 Transmit message buffer access locking for both M-CAN and TT-CAN

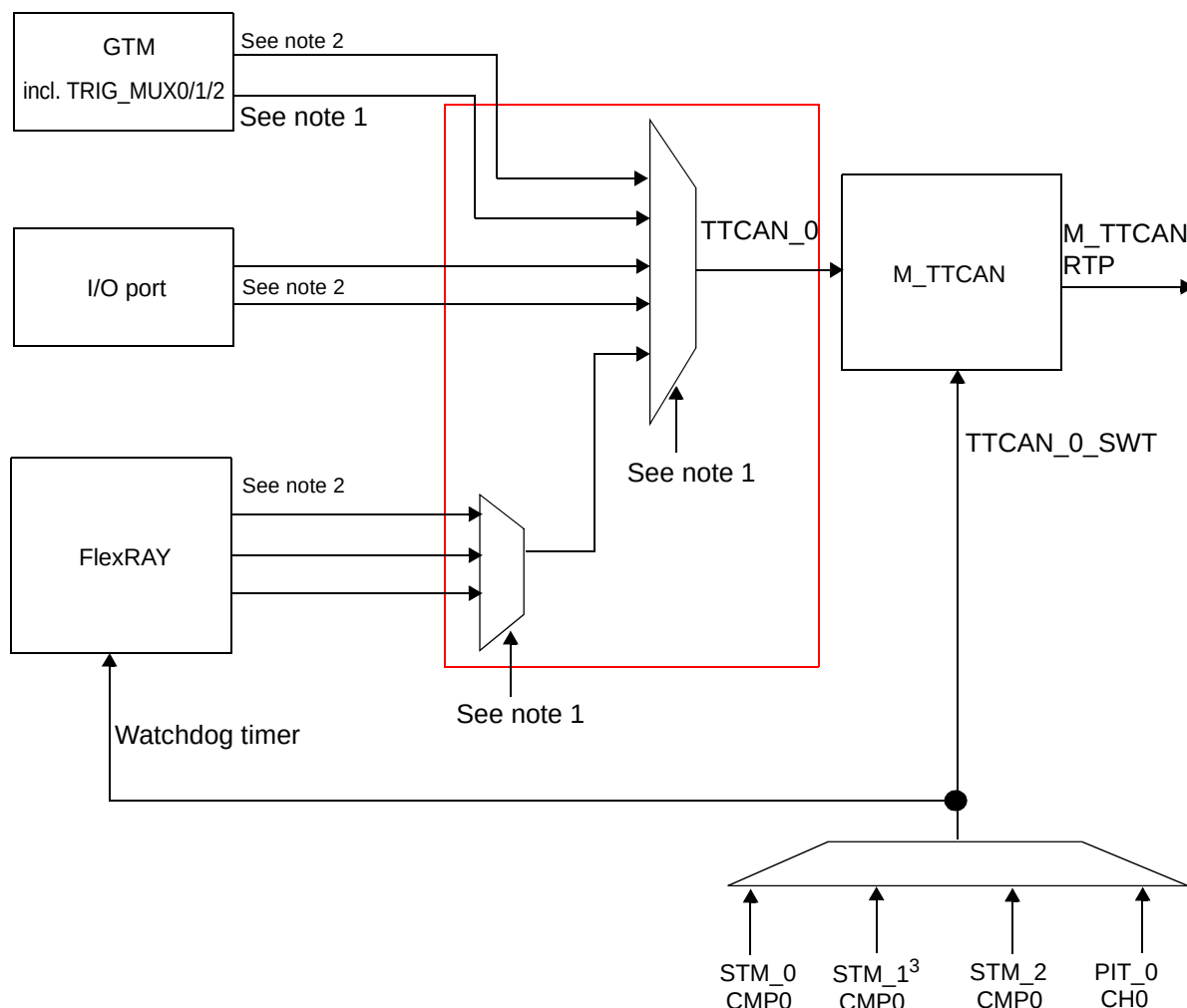
Each CAN module can have at most 32 TX Buffers and the size of each TX Buffer is 4 words. The CPU write locking mechanism ensures that whenever any TX Buffer's pending request is set (by the TRP bit of the TXBRP register), the CPU does not get any write access to the corresponding TX buffers. If the CPU tries to write into a TX Buffer whose pending request is set it receives an error response on the CAN RAM Transfer Error port.

46.6.3.2 Trigger memory element access locking for TT-CAN

Up to 64 trigger memory elements can be configured in a TT-CAN Module and the size of each trigger memory element is 2 words. The trigger memory elements have to be written when the TT-CAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed. The CPU write locking mechanism ensures that whenever the TT-CAN is not in the INIT state (CCE bit of the CCCR register is set to '0'), the CPU does not get any write access to the Trigger Memory Elements. If the CPU tries to write into a TX Buffer, while the TT-CAN is not in the INIT state, it will get an error response on the CAN RAM Transfer Error.

46.7 Time Triggered CAN matrix and basic cycles management

The following figure shows hardware control for M_TTCAN cycles management.



Note 1: Multiplexers and control logic is implemented in the SIUL block. See SIUL Multiplexed Signal Configuration Register (SIUL2: MSCR 753, MSCR 754).

Note 2: For the signal details from various blocks see section SIUL Module-port MSCR assignments and SSS values (SIUL2: MSCR 753, MSCR 754).

Note 3: See device configuration chapter for the availability of the STM instances.

Figure 686. Hardware control for M_TTCAN cycles management

46.8 CAN nodes 1 and 2 I/Os sharing

Each CAN node has its Tx/Rx signals connected to separated GPIO pairs. It is required for M_CAN_1 and M_CAN_2 to share one common GPIO pair. This feature is described below.

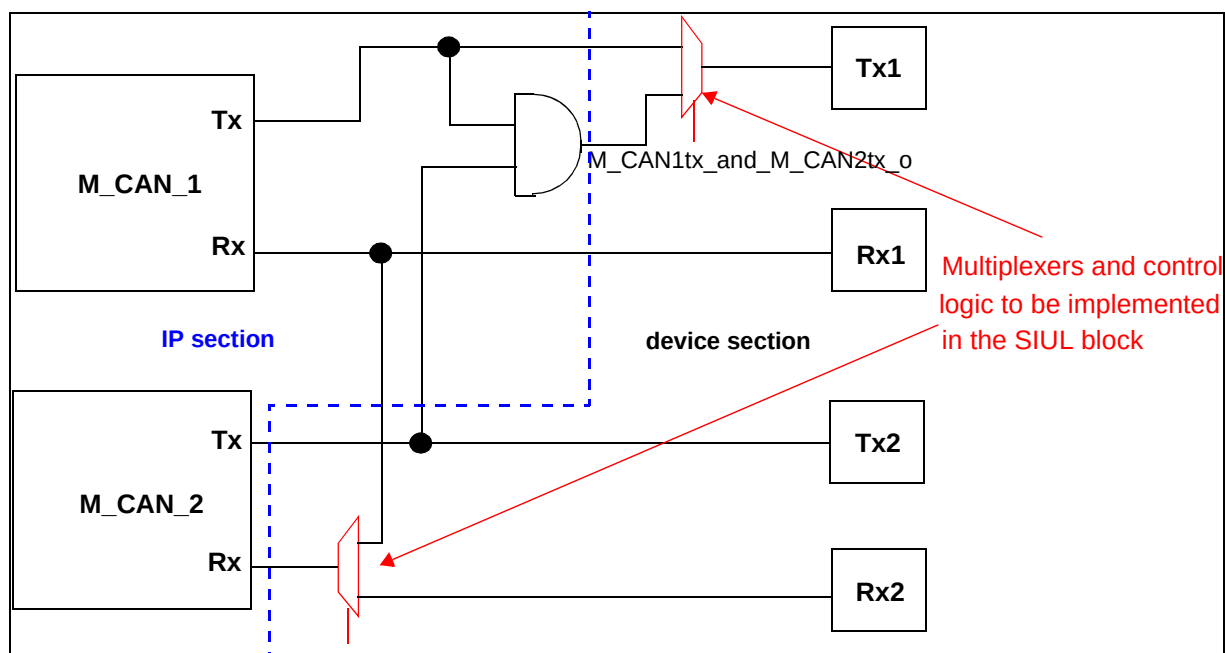


Figure 687. M_CAN_1 and M_CAN_2 connections to GP I/Os

46.9 External signal description

The CAN subsystem has following external pins.

- M_CAN_x_RX: M_CAN x receive input
- M_CAN_x_TX: M_CAN x transmit output
- M_TTCAN_x_RX: M_TTCAN x receive input
- M_TTCAN_x_TX: M_TTCAN x transmit output

46.10 Memory map and register description

46.10.1 Shared memory map

[Table 756](#) shows the shared memory map vs CAN subsystem transmit, receive and filters elements.

Table 756. Shared memory map vs CAN transmit, receive and filters elements

Start Address Offset (byte)	End Address Offset (byte)	CAN block/sub block		Comment
0x0000	0x0CFF	M_CAN1	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer according to Table 754 .
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 1	
			Rx Buffer	
			Tx Event FIFO	
0x0D00	0x0EFF		Tx Buffers	Protected from CPU write access using the M_CAN1 TXBRP register output
0x0F00	0x1BFF	M_CAN2	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer according to Table 754 .
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 1	
			Rx Buffer	
			Tx Event FIFO	
0x1C00	0x1DFF		Tx Buffers	Protected from CPU write access using the M_CAN2 TXBRP register output
0x1E00	0x2AFF	M_TTCAN0	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer according to Table 755 .
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 1	
			Rx Buffer	
			Tx Event FIFO	
0x2B00	0x2CFF		Tx Buffers	Protected from CPU write access using the M_TTCAN0 TXBRP register output
0x2D00	0x2EFF		M_TTCAN triggers	Protected from CPU write access using the M_TTCAN0 CCE register output
0x2F00	0x3BFF	M_CAN3	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer according to Table 754 .
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 1	
			Rx Buffer	
			Tx Event FIFO	
0x3C00	0x3DFF		Tx Buffers	Protected from CPU write access using the M_CAN3 TXBRP register output

Note: There is no write access protection from a CAN module to the message memory space. The write access protection is only for CPU access to the message memory.

Note: The start address for every memory section is to be configured by the user in the relevant CAN modules registers. However there is no hardware check in the RAM interface for the start address consistency.

However the number of filters, Tx/Rx buffers, triggers used, it is mandatory for the user to configure the start address registers, specified in [Table 757](#), in order to fit the memory mapping specified in [Table 756](#).

Note: The CPU uses the peripheral interface to access the memory locations. It must be noted that there is no transfer error reported to the CPU in case of access to reserved memory locations.

Table 757. Detailed mapping in CAN subblocks

Address offset (byte)	CAN subblock	Element name	Comment
0x0000 --- 0x01FF	Standard Filters	Standard Message ID filter 0 --- Standard Message ID filter 127	address configured in SIDFC [FLSSA]
0x0200 --- 0x03FF	Extended Filters	Extended Message ID filter 0 --- Extended Message ID filter 63	address configured in XIDFC [FLESA]
0x0400 --- 0x07FF	Rx FIFO0	Rx FIFO0 element 0 --- Rx FIFO0 element 63	address configured in RXF0C[F0SA]
0x0800 --- 0x0BF3	Rx FIFO1	Rx FIFO1 element 0 --- Rx FIFO1 element 63	address configured in RXF1C[F1SA]
0x0BF4 ... 0x0BFF	Rx Buffer	Rx Buffer element 0 --- Rx Buffer element 63	address configured in RXBC[RBSA]
0x0C00 --- 0x0CFF	Tx Event FIFO	Tx event FIFO element 0 --- Tx event FIFO element 31	address configured in TXEFC[EFSA]
0x0D00 --- 0x0EFF	Tx Buffers	Tx Buffer element 0 --- Tx Buffer element 31	address configured in TXBC[TBSA]
0x0F00 --- 0x10FF	M_TTCAN triggers (on M_TTCAN0 only)	Trigger element 0 --- Trigger element 63	address configured in TMC[TMSA]

47 Serial Interprocessor Interface (SIPI)

47.1 Introduction

The Serial Interprocessor Interface (SIPI) is an application layer protocol that runs on top of the LFAST module. It is used by the local device to access the shared memory of a remote device. SIPI defines point-to-point full duplex communication between two devices, where LFAST works as a physical medium of communication between both the devices.

47.1.1 Scalability

The SIPI protocol is designed to provide a sophisticated, high bandwidth, multimaster, multi-channel memory interface between two devices using few interconnecting signals. But the protocol is designed in such a way that a subset of the protocol can be implemented, where die area is more important than features.

Main scalable features:

- Number of concurrent channels:
 - Full Implementation = 4
 - Minimum Implementation = 1
- Full implementation has a node both as Initiator and Target of commands. Minimum Implementation either Initiator or Target
- Full Implementation includes a block transfer feature, but this feature is optional.

The rest of this section describes a full implementation of SIPI, which includes: an Advanced High-Performance Bus (AHB-Lite) master interface, Direct Memory Access (DMA) interface, LFAST Tx/Rx (transmit/receive) interfaces along with Peripheral Bus Interface (IPS).

47.2 Overview

An instance of SIPI can act as initiator, or target or both. SIPI can access the shared memory directly through its AHB master interface or through its DMA interface. DMA interface is used when the node acts as an initiator while the AHB Master interface will be used when the node acts as target. SIPI has four channels, with one channel (Channel 2) having data streaming capability. Payload width for channel 0, 1 and 3 is 32 bits, and for channel 2 data widths can be 32 bits, or 256 bits when streaming. Any of these channels can be used for DMA access or bus interface access depending on the setting of the CCR*n*. CRC encoder calculates the CRC on the command frame. Then the SIPI initiator appends the CRC to the end of the frame before transmission.

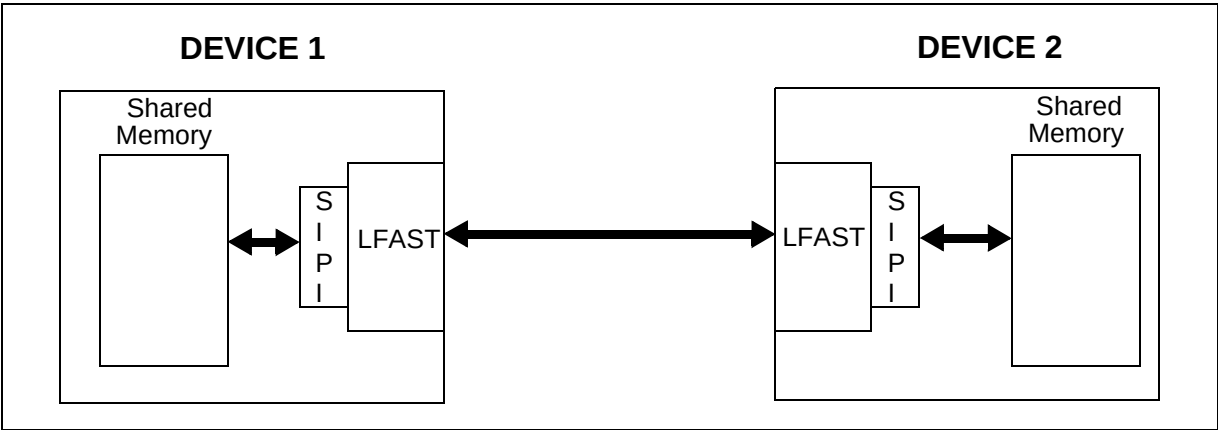


Figure 688. Interprocessor communication diagram

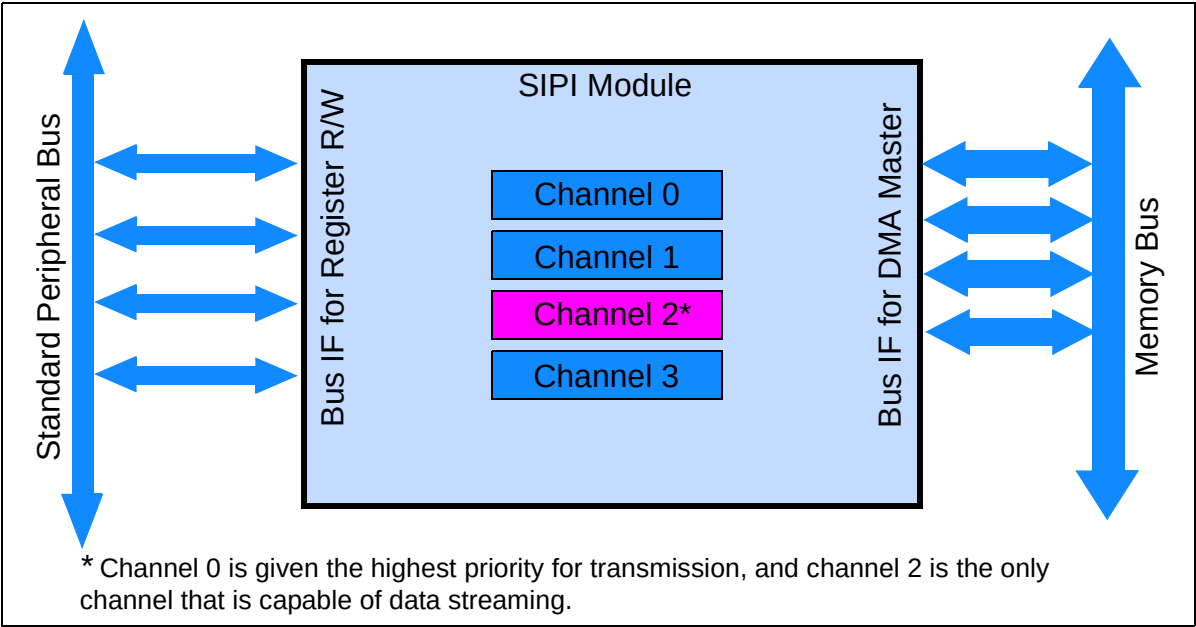
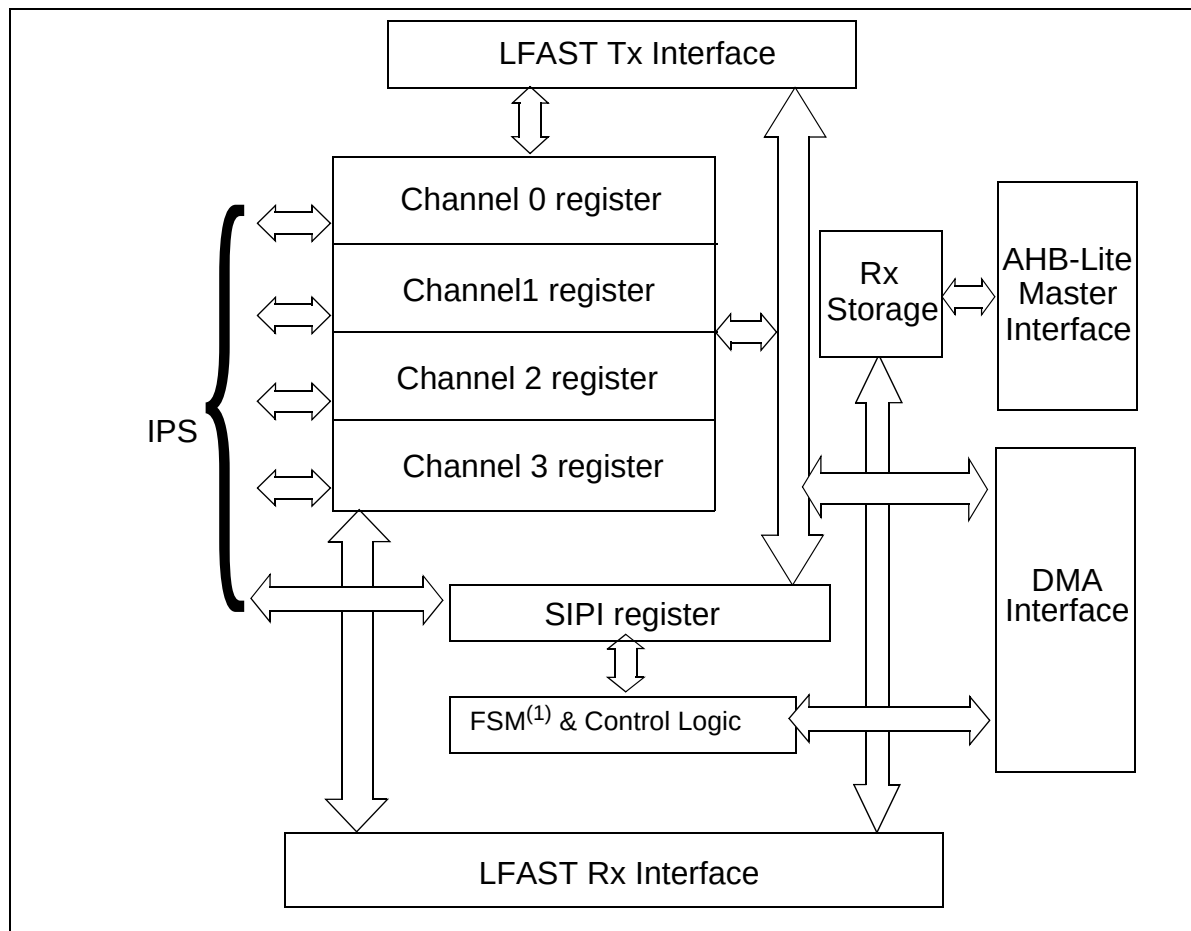


Figure 689. SIPI module

47.3 SIPI block diagram



1. Finite State Machine

Figure 690. SIPI block diagram

47.4 Feature description

This section describes the features of the SIPI module.

47.4.1 Main Features

- Point-to-point communication between two devices
- Full duplex communication
- Four channels, including one channel with data streaming capability
- Configurable DMA access for each channel
- CRC protection mechanism
- Timeout protection mechanism
- Fixed priority channel selection
- Data size up to 256 bits on streaming channel
- Common tag pool for assigning sequential transfer IDs to every new transfer
- AHB master interface that is used by target node to access shared memory
- Target node contains a set of nine 32-bit internal registers to store commands
- Up to two outstanding requests are supported at initiator.

47.4.2 Standard features

- IPS bus interface
- SPP DMA2x bus interface
- AHB Master Interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

47.5 Functional description

47.5.1 External signals

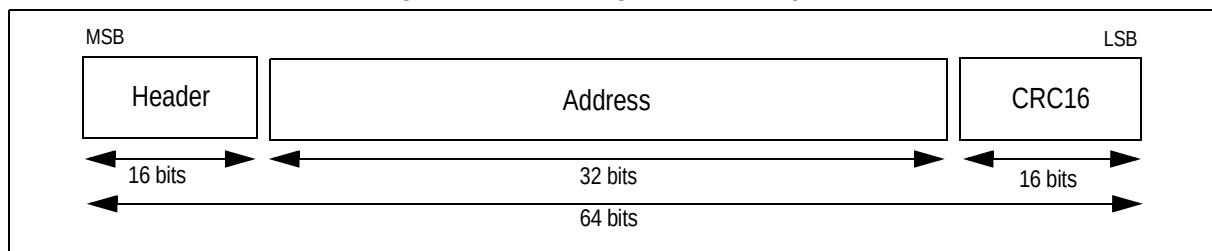
The SIPI has no chip external signals.

47.5.2 Frame format

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

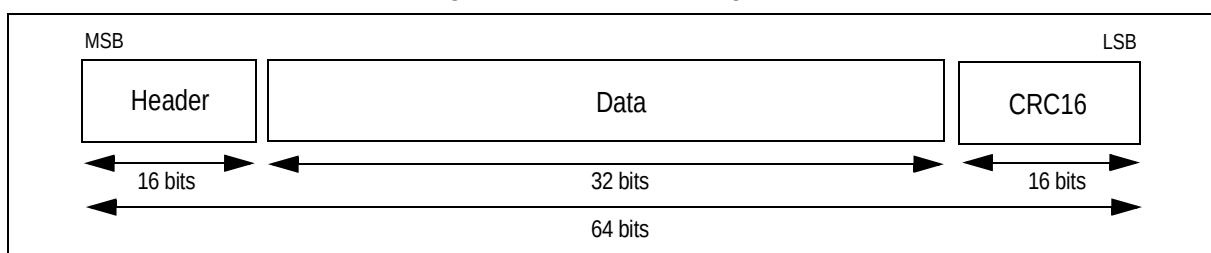
47.5.2.1 Register read request

Figure 691. SIPI register read request



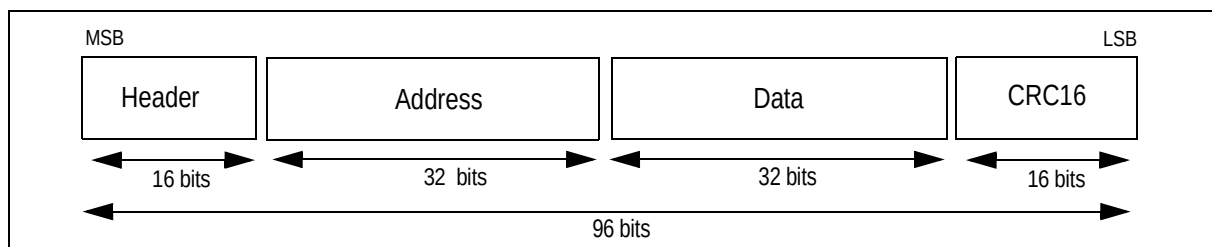
47.5.2.2 Register read response, ID request response

Figure 692. SIPI read response



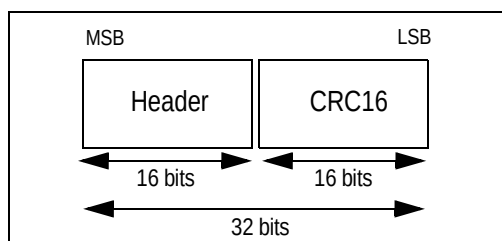
47.5.2.3 Register write request

Figure 693. SIPI register write request



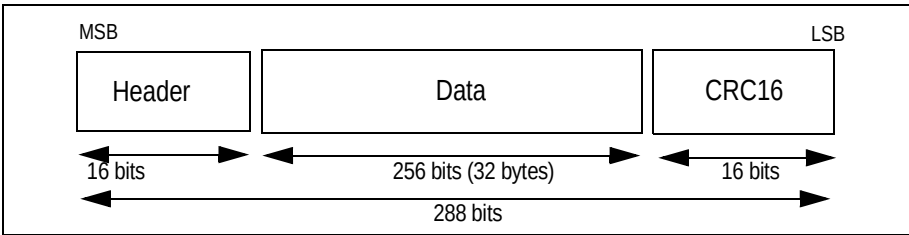
47.5.2.4 Trigger transfer, ID transfer, write acknowledge and streaming write acknowledge

Figure 694. SIPI write acknowledge



47.5.2.5 Streaming write request

Figure 695. Streaming write request format^{(1),(2)}



- 1. Direct write operations are used to set the streaming address.
- 2. Streaming data write is performed using the format shown in [Figure 695](#).

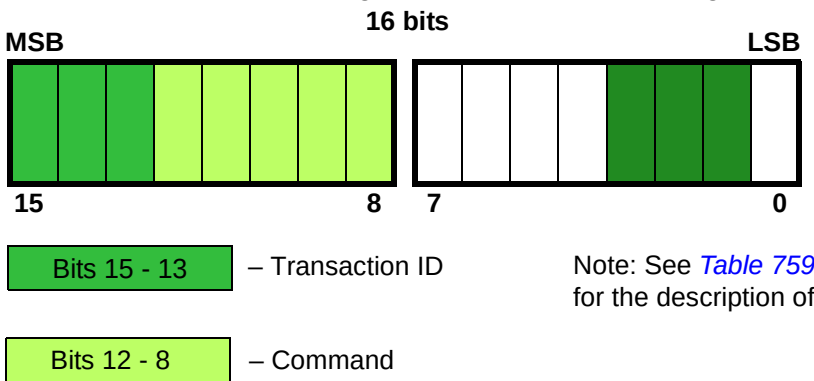
47.5.2.6 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

47.5.2.6.1 SIPI header coding

[Figure 696](#) shows the SIPI header coding.

Figure 696. SIPI header coding



[Table 758](#) shows the command coding for the bits[12:8] of the header.

Table 758. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
00	Read 8 bits	64
01	Read 16 bits	64
02	Read 32 Bits	64
03	Reserved	—
04	Write 8 bits with ACK	96
05	Write 16 bits with ACK	96
06	Write 32 bits with ACK	96

Table 758. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
07	Reserved	—
08	ACK – OK	32
09	ACK – Fault	32
0A	Read Answer – OK	64
0B	Reserved	—
0C	Trigger comm and with ACK	32
0D	Reserved	—
...	...	...
11	Reserved	—
12	ID Register Read Request	32
13	Reserved	—
...	...	...
16	Reserved	—
17	Stream Data with ACK (32 bytes)	288
18	Reserved	—
...	...	...
1F	Reserved	—

Figure 697 shows the contents of bits[7:0] in the SIPI header.

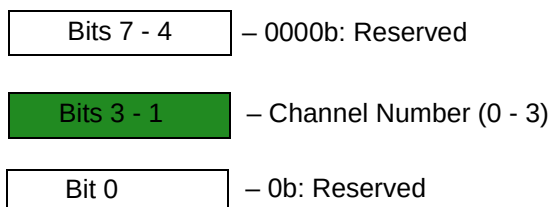


Figure 697. SIPI header channel field

Table 759 shows the channel number coding.

Table 759. SIPI Header Channel Number Coding

SIPI Channel Name	Channel Number Coding in header(s)		Comment
	SIPI Channel Coding Select Bit		
	Code Table I (1)	Code Table II (0)	
0 (Channel A)	000b	100b	In use
1 (Channel B)	001b	101b	In use

Table 759. SIPI Header Channel Number Coding

SIPI Channel Name	Channel Number Coding in header(s)		Comment
	SIPI Channel Coding Select Bit		
	Code Table I (1)	Code Table II (0)	
2 (Channel C)	010b	110b	In use
3 (Channel D)	011b	111b	In use
4 (Channel E)	100b	000b	Reserved
5 (Channel F)	101b	001b	Reserved
6 (Channel G)	110b	010b	Reserved
7 (Channel H)	111b	011b	Reserved

47.5.2.7 Address field

The address field is 32 bits wide with the MSB transmitted first.

47.5.2.8 Payload field

[Table 760](#) shows the payload sizes of the LFAST frame.

Table 760. Payload size of LFAST channel frame

LFAST Code (bin)	SIPI Code (bin)	LFAST Payload Size (bits)	(bytes)
000	—	8	1
001	—	32	4
010	010	64	8
011	011	96	12
100	100	128	16
101	101	256	32
110	110	512	64
111	111	288	36

[Table 761](#) shows the converted coding of LFAST for a given SIPI code.

Table 761. Converted coding of LFAST channel code for SIPI headers

LFAST Code (bin)	SIPI Code (bin)	Channel (hex) ⁽¹⁾
0100	100	A
0101	101	B
0110	110	C
0111	111	D

Table 761. Converted coding of LFAST channel code for SIPI headers (continued)

LFAST Code (bin)	SIPI Code (bin)	Channel (hex)⁽¹⁾
1000	000	E (not used by SIPI)
1001	001	F (not used by SIPI)
1010	010	G (not used by SIPI)
1011	011	H (not used by SIPI)

1. SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
 SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
 SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
 SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

47.5.2.9 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome ($x^{16} + x^{12} + x^5 + 1$). The data stream is transmitted with the MSB sent first.

47.5.2.9.1 CRC examples

47.5.2.9.1.1 Example 1 – 32-bit write on channel 1 with ID1

- Header = 260Ah
- Address = 1122_3344h
- Data = CCDD_EEFFh
- CRC = BF7Dh

47.5.2.9.1.2 Example 2 – 32-bit read on channel 2 with ID2

- Header = 420Ch
- Address = 89AB_CDEFh
- CRC = 6B80h

47.5.2.9.1.3 Example 3 – Event command on channel 3 with ID3

- Header = 6C0Eh
- CRC = B286h

47.6 Transfer types

This section describes the available transfer types of the SIPI module.

47.6.1 Read transfer

A Read transfer can be of two types:

- Read request transfer (at initiator node)
- Read response transfer (at target node)

47.6.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in [Figure 698](#).

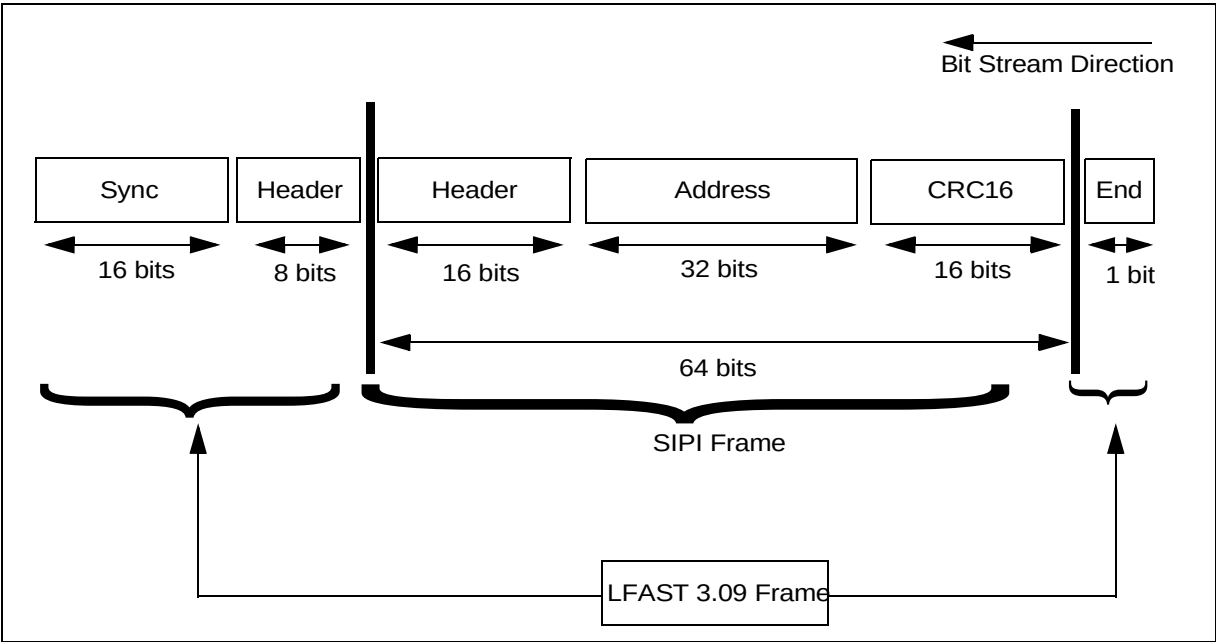


Figure 698. Read request transfer

47.6.2 Register read answer transfer

In response to a read request by the initiator node, the target node will send header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (see [Figure 699](#)). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (see [Figure 700](#)).

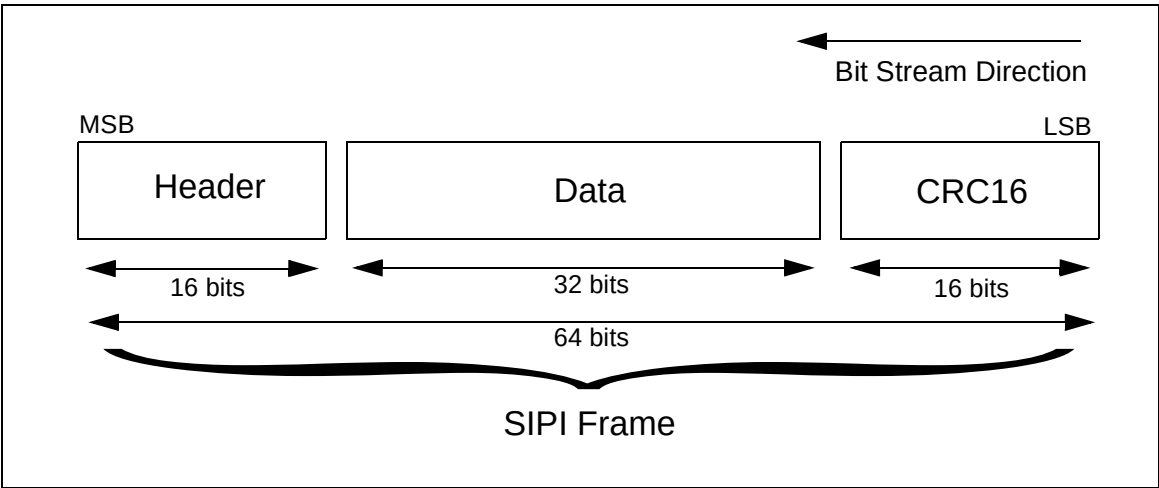
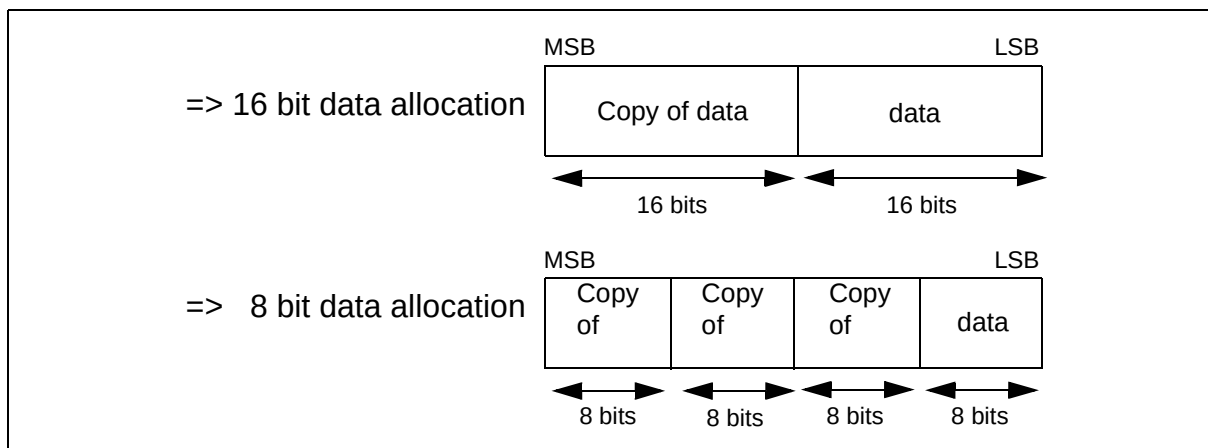


Figure 699. Read answer transfer

**Figure 700. Data allocation**

Note: In case of 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
 Bits[1:0]
 00 - byte 3 enabled (MSB)
 01 - byte 2 enabled
 10 - byte 1 enabled
 11 - byte 0 enabled (LSB)

In case of 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable.
 Bit[0]
 0 - half-word 1 enabled (MSB)
 1 - half-word 0 enabled (LSB)

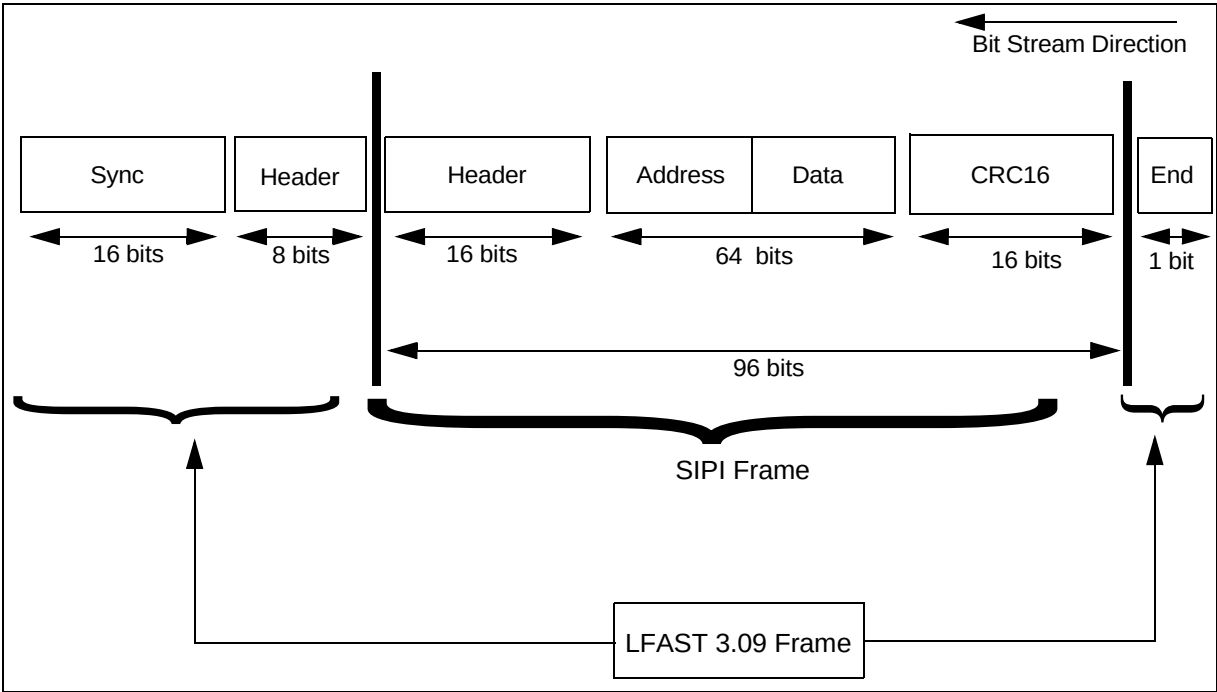
47.6.3 Register write transfer

Register Write transfer can be of two types:

- Normal write transfer - channels 0, 1, 2 and 3
- Streaming data transfer - channel 2 only

Register Write transfer can be done through a normal write transfer - channels 0, 1, 2 and 3 as shown in [Figure 701](#).

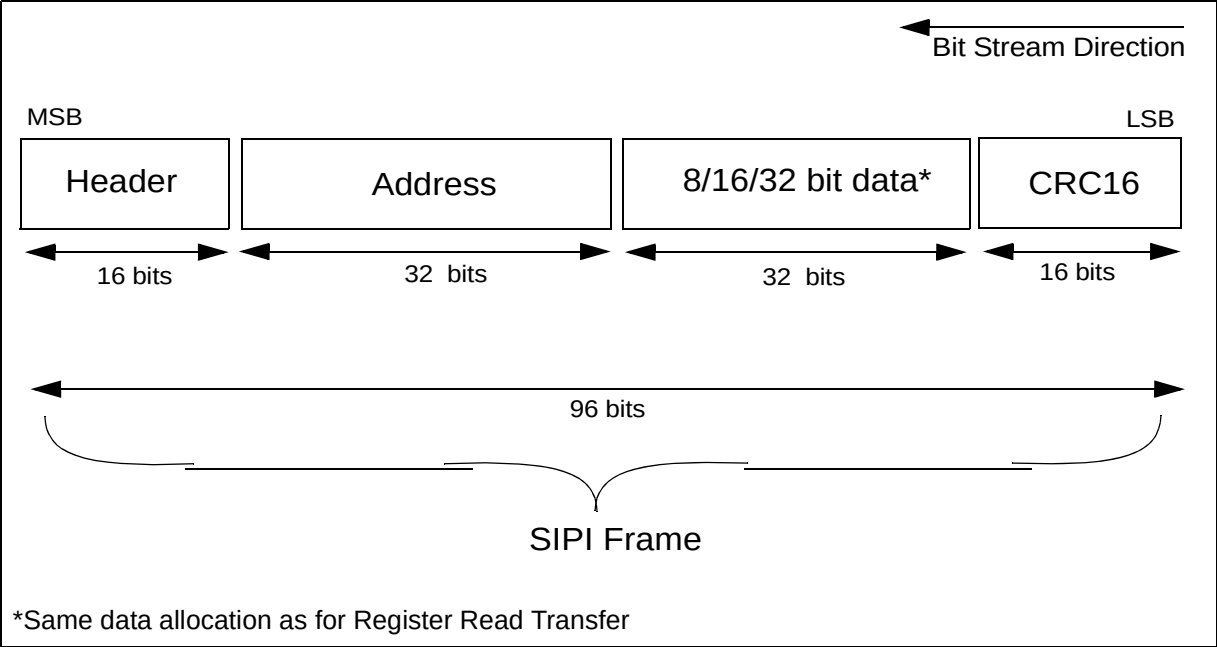
Figure 701. Register write transfer



47.6.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and 16-bit CRC as shown in [Figure 702](#).

Figure 702. Write transfer



Note: In the case of an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as a byte enable.

- Bits[1:0]:
- 00 – byte 3 enabled (MSB)
- 01 – byte 2 enabled
- 10 – byte 1 enabled
- 11 – byte 0 enabled (LSB)

In the case of a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as a half-word enable.

- Bit[0]:
- 0 – half-word 1 enabled (MSB)
- 1 – half-word 0 enabled (LSB)

47.6.3.2 Streaming write transfer

Streaming write transfers have 256 bits of payload.

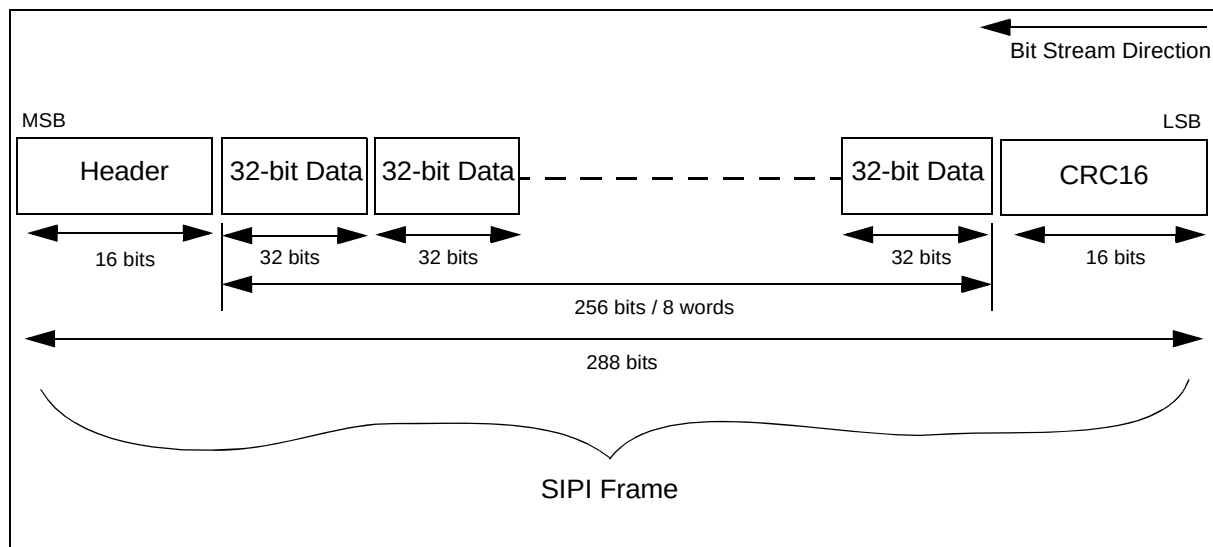
47.6.3.2.1 Streaming write transfer with address

Setting the address is performed using a direct write transfer. MAXCR and SIPIARR are written before the SIPIACR. This is to ensure that the unwanted behavior of the SIPI trying to access non-shared memory is avoided. This is only true for the very first streaming command, subsequent streaming command(s) may not have the need to write MAXCR and SIPIARR.

47.6.3.2.2 Streaming transfer with data

[Figure 703](#) shows the packet structure of streaming transfers that contain data.

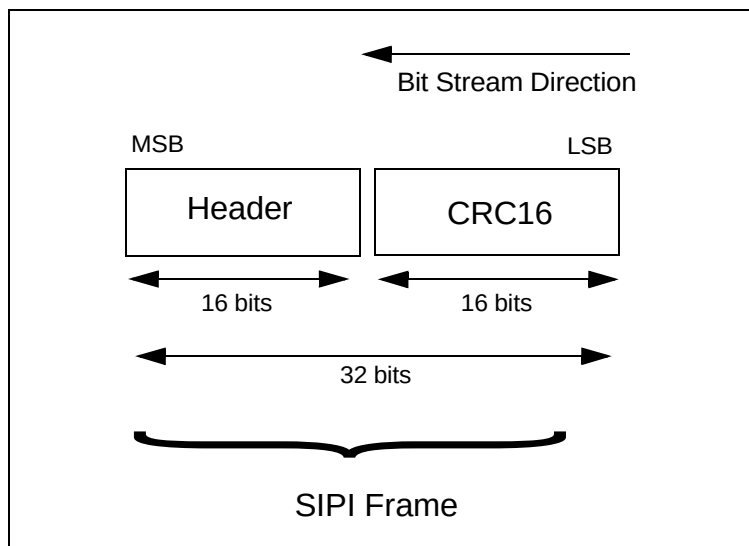
Figure 703. Streaming transfer with data



47.6.4 Write Acknowledge transfer

A Write Acknowledge transfer contains only header and CRC bits (see [Figure 704](#)). The CRC bits are calculated on the header field.

Figure 704. Write Acknowledge transfer

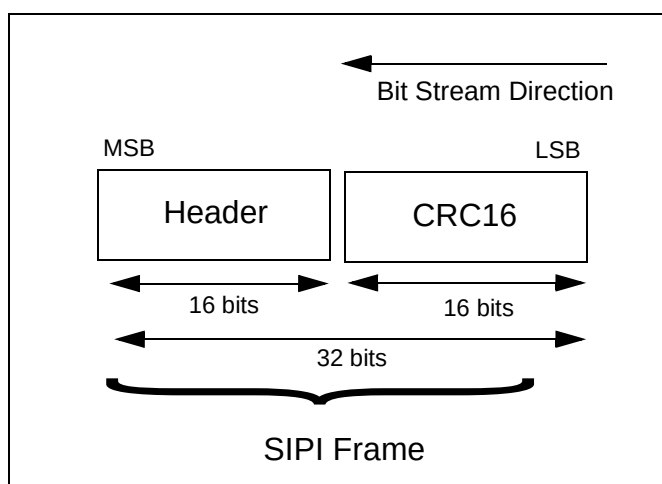


47.6.5 ID request response

47.6.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in [Figure 705](#).

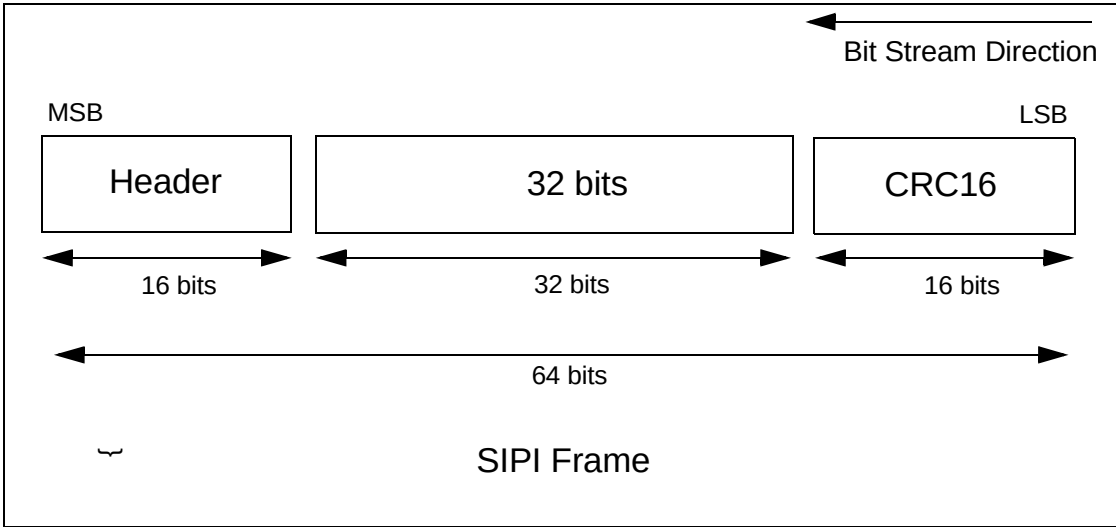
Figure 705. ID request transfer



47.6.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in [Figure 706](#).

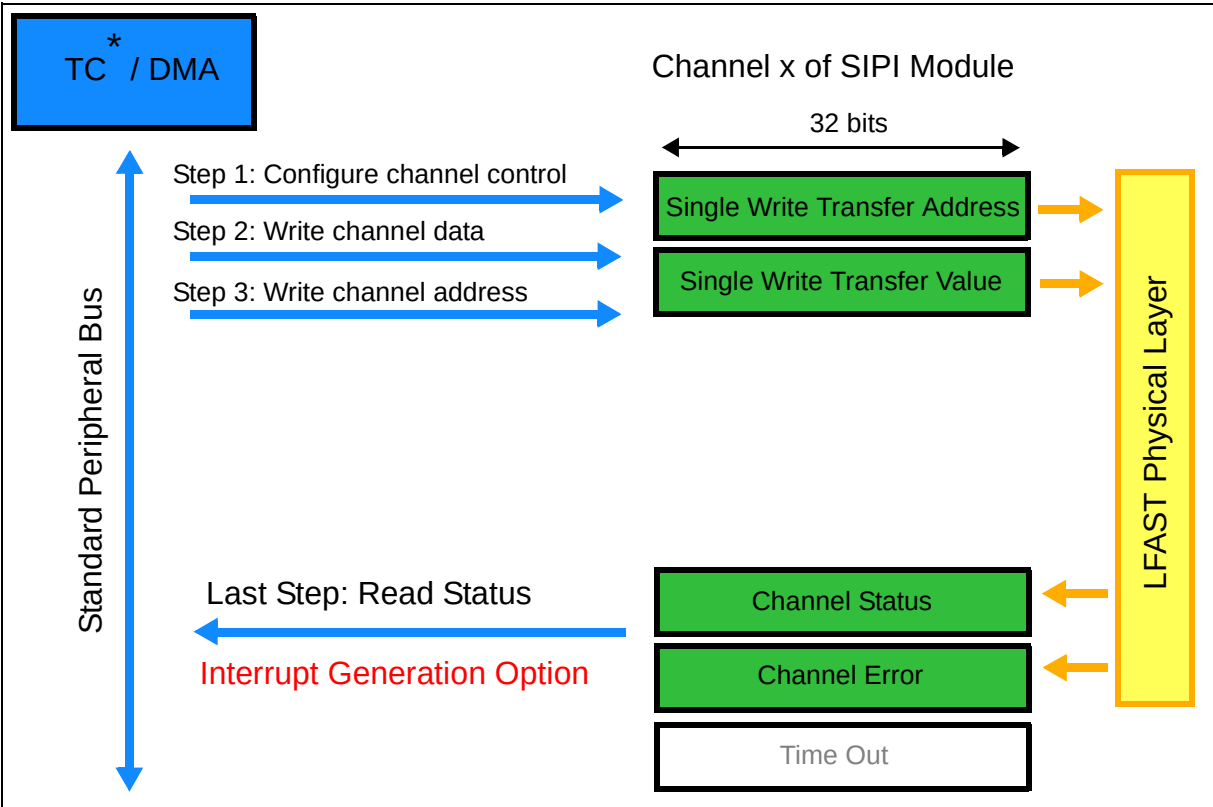
Figure 706. ID request response transfer



Note: The ID request response contains the value of the CHIP ID in place of data.

47.7 Transfer API and flow charts

Figure 707. SIPI single register write API



*Note: TC = Transmit Command

For a single Write Transfer Request generation (*Figure 707*):

1. Configure data and CCR n at the initiator node.
2. Configure CAR n at the initiator node.

As soon as the channel address register (CAR n) is written and if CCR n [CHEN] = 1, the initiator SIPI will calculate the CRC on header, address and data field and will start transmitting data to LFAST.

3. Software polls the status register bits (CSR n) to determine when the request completes, CSR n [ACKR] = 1. An interrupt will be generated if the corresponding CIR n [ACKIE] = 1.

On a single Write transfer request reception (*Figure 708*):

1. Target node will place the address, data and control information on its AHB master interface.
2. When the process is completed, the target node will generate an acknowledge frame and send it back to the LFAST.

Figure 708. SIPI Single Register Write API – Flow Chart

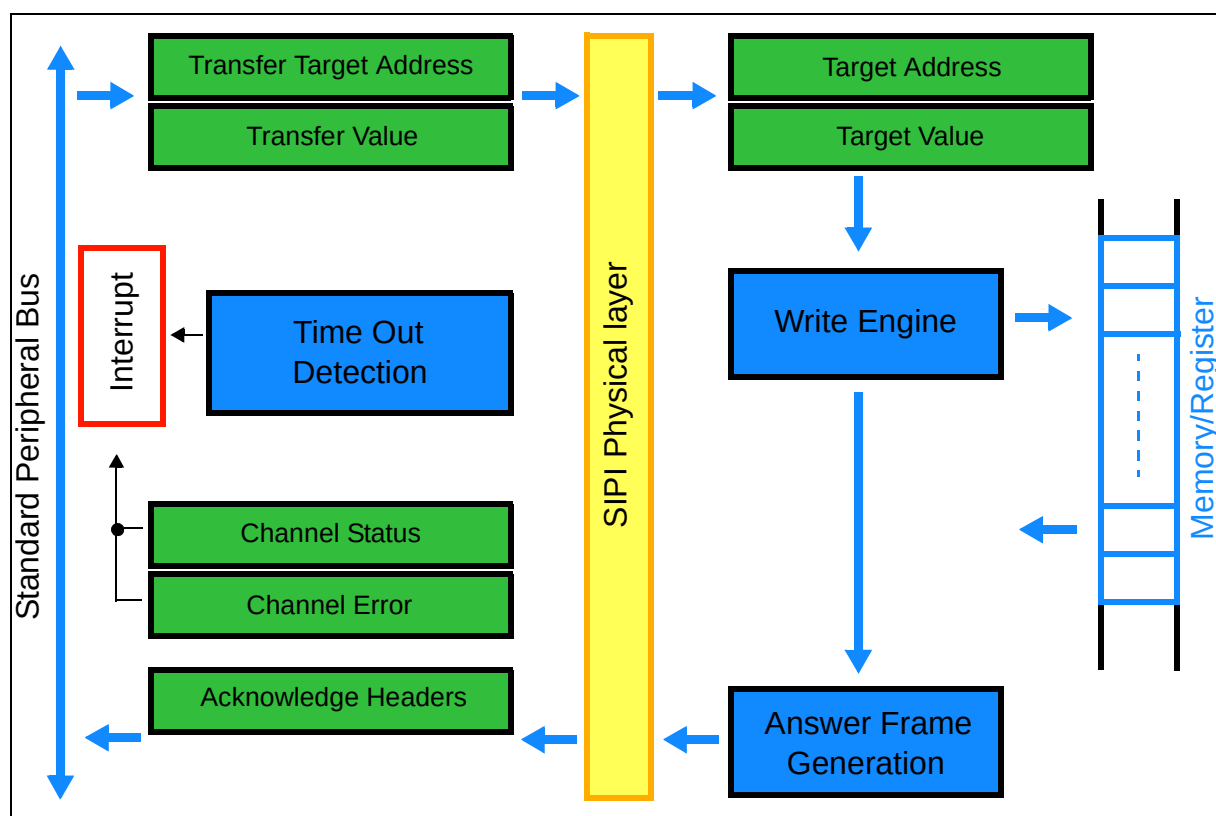
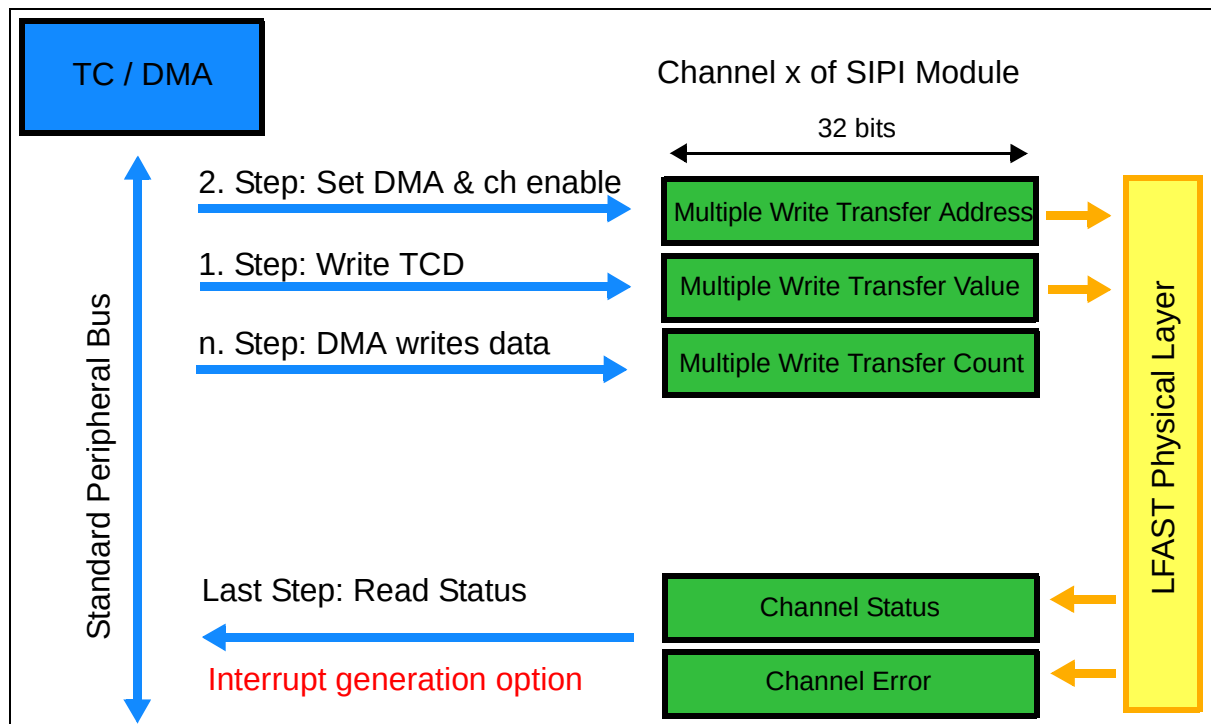


Figure 709. SIPI Multiple Register Write API



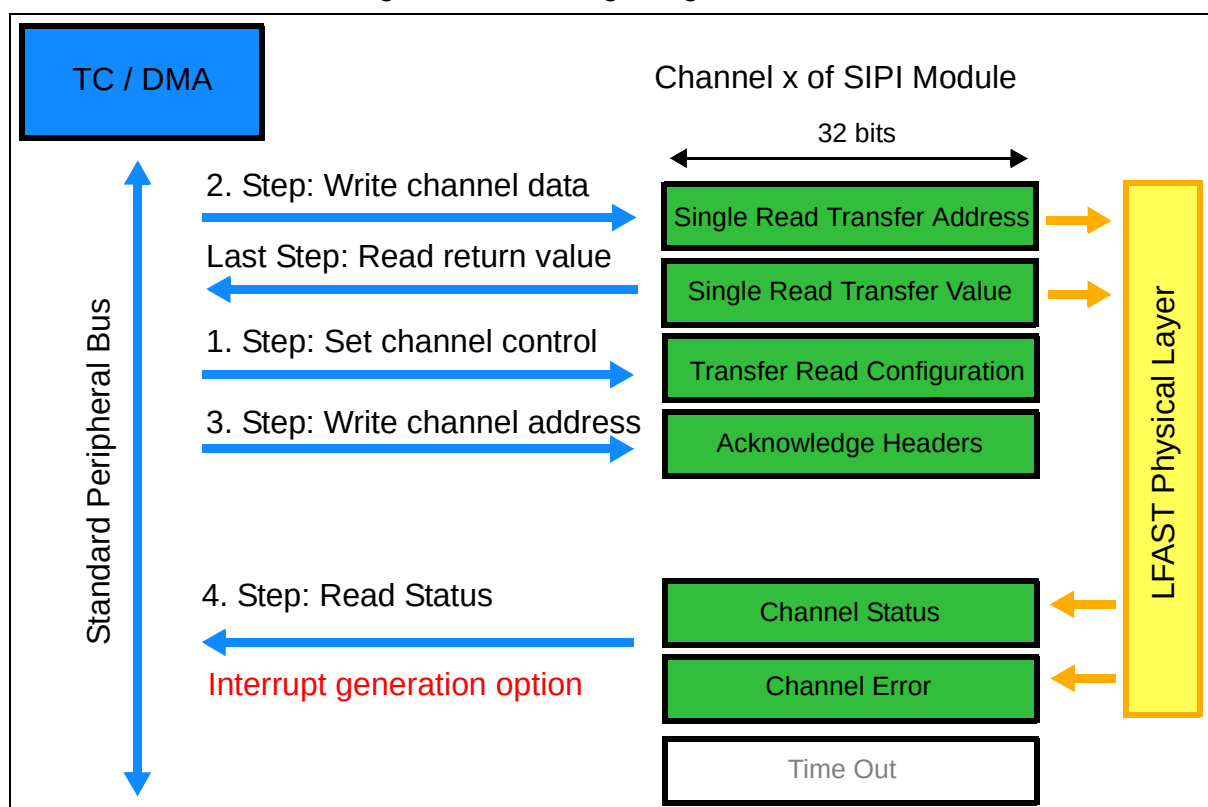
For Multiple Write Transfer Request generation ([Figure 709](#)):

1. Software will configure the Transfer Control Descriptor (TCD) of the DMA.
2. Software will write $CCR_n[CHEN] = 1$ and $CCR_n[DAN] = 1$.
3. SIPI will generate `ipd_req` and send it to the DMA controller.
4. SIPI will start copying data into the CDR_n through its DMA interface, depending on the transfer count and data registers size. $CDR2[0]$ should be written with the MSB.
5. When the copying process is complete, initiator SIPI will calculate CRC on header, address and data field and start transmitting data to LFAST.
6. Software should poll the CSR_n status register bits to determine if the request has completed. If $CSR_n[ACKR] = 1$, an interrupt will be generated (if the corresponding $CIR_n[ACKIE] = 1$).
7. If $CCR_n[DEN] = 1$, the SIPI request will transfer to the DMA controller. If not, the `ipd_req` will not be sent and the state machine goes idle.
8. Steps 4–7 will be repeated.

On Multiple Write transfer request reception ([Figure 709](#)):

1. Target node will place the address, data and control information on its AHB Master Interface.
2. When the process is completed, target node will generate an acknowledge frame and send it back to the LFAST.

Figure 710. SIPI Single Register Read API



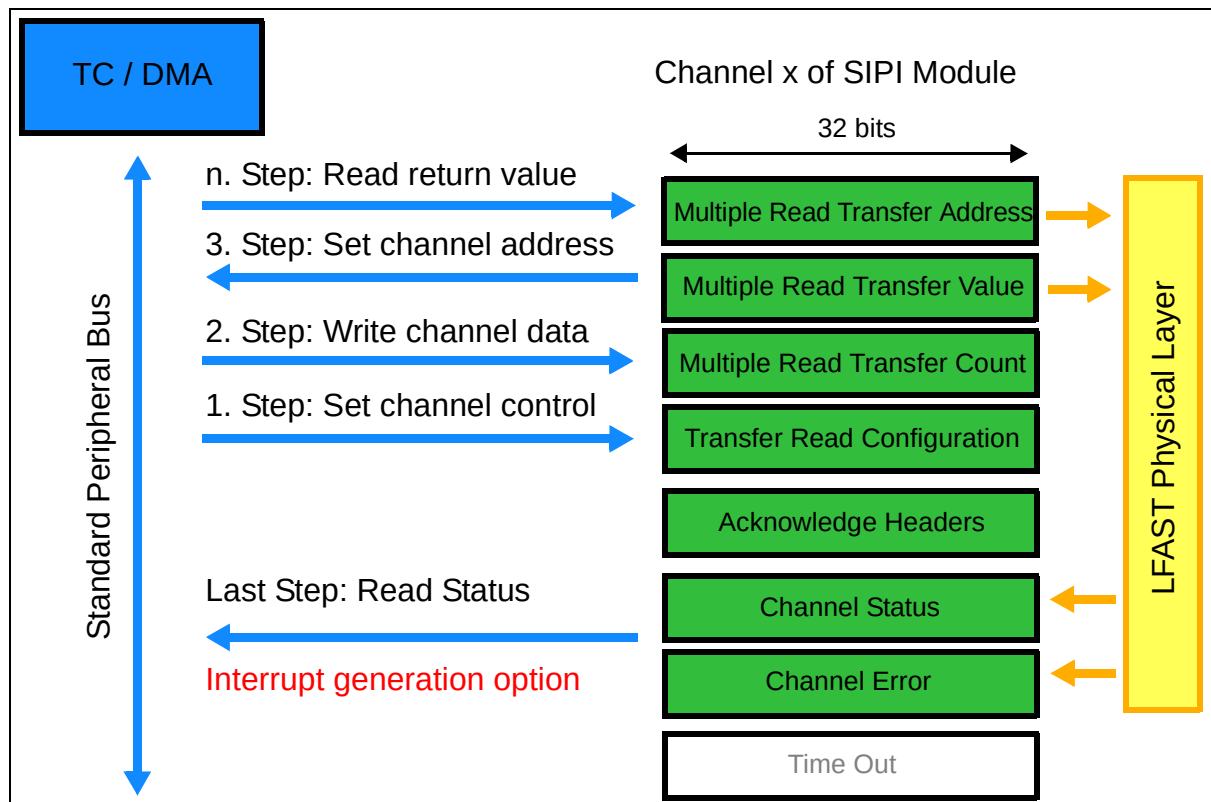
For Single Read Transfer Request generation ([Figure 710](#)):

1. Software/DMA will configure $CCRn$ and $CDRn$, with the last step by writing $CARn$.
2. As soon as CAR is written, the initiator SIPI will calculate CRC of the header and address fields and start transmitting data to the LFAST.
3. Software should poll $CSRn$ to determine when the request has completed. If $CSRn[RAR] = 1$, and interrupt will be generated (if $CIRn[RAIE] = 1$). If $CSRn[RAR]$ does not set then $SIPERR[TOEn] = 1$, which indicates a timeout has occurred.
4. If $CSRn[RAR] = 1$, then software can read the data register, or can take other necessary action if $CSRn[RAR] = 0$.

On Single Read transfer request reception ([Figure 710](#)):

1. Target node will place the address and control information on its AHB Master Interface.
2. When the process is completed, target node will send read response back to LFAST.

Figure 711. SIPI multiple register read API



For multiple read transfer request generation ([Figure 711](#)):

1. Software/DMA will configure $CCRn$ and $CDRn$, with the last step by writing $CARn$.
2. As soon as channel address register is written, initiator SIPI will calculate CRC on the header and address fields, then start transmitting data to the LFAST.
3. Software should poll $CSRn$ to determine when the request has completed. If $CSRn[RAR] = 1$ and $CIRn[RAIE] = 1$, an interrupt will be generated. If $CSRn[RAR]$ does not set, then $SIPERR[TOEn] = 1$, which indicates a timeout.
4. Steps 2–3 will be repeated for multiple requests.

On Multiple Read transfer request reception ([Figure 711](#)):

1. Target node then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

[Figure 712](#), [Figure 713](#), [Figure 714](#) and [Figure 715](#) show the SIPI data streaming of data.

Figure 712. SIPI data stream model

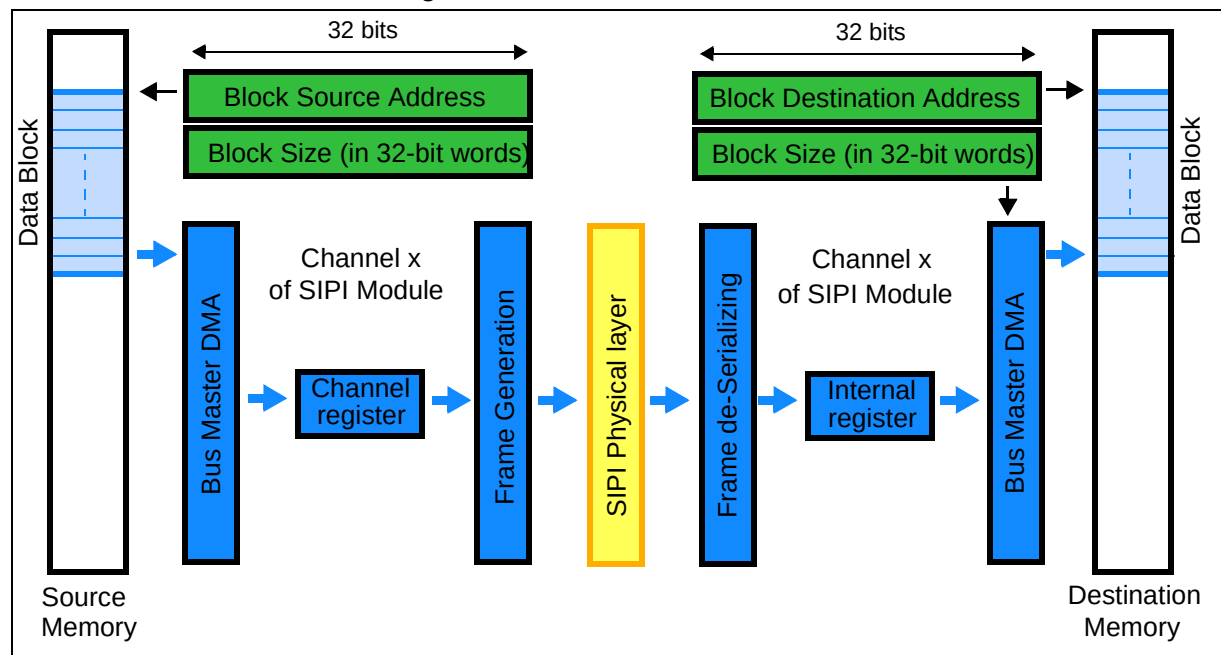


Figure 713. SIPI data stream model - Initiator

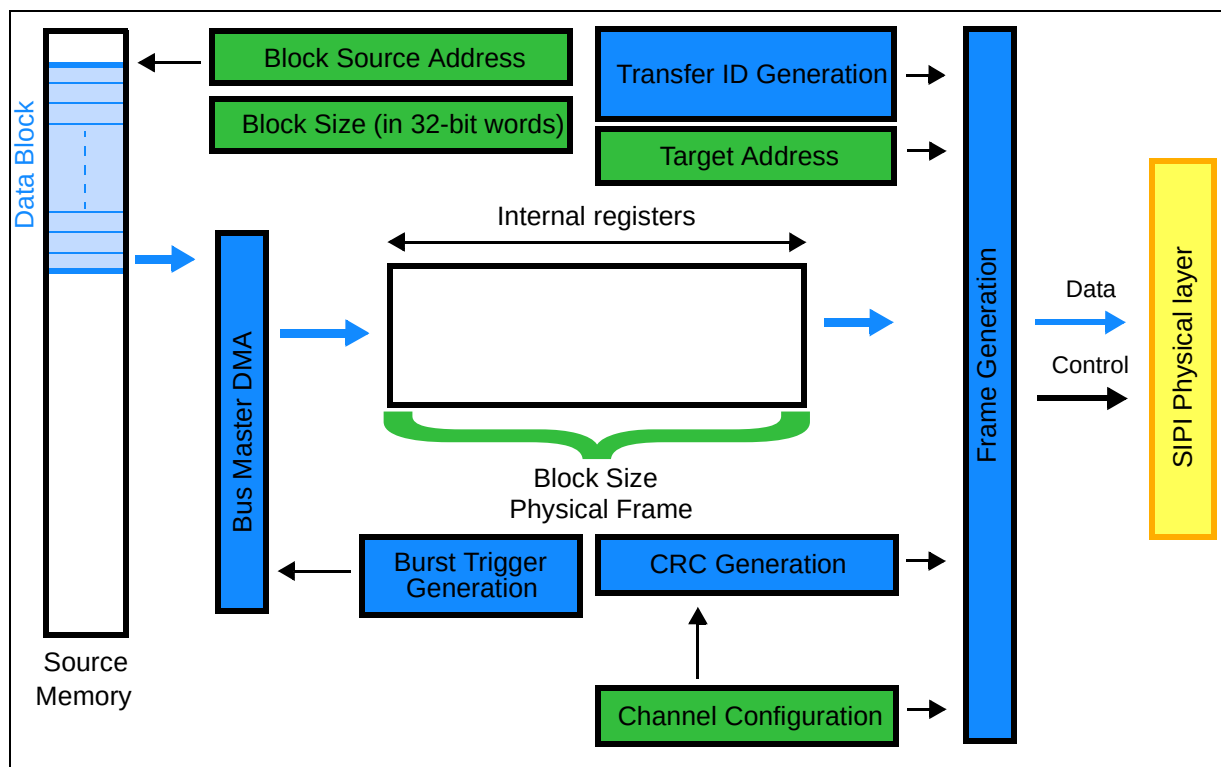


Figure 714. SIPI data stream model - Target

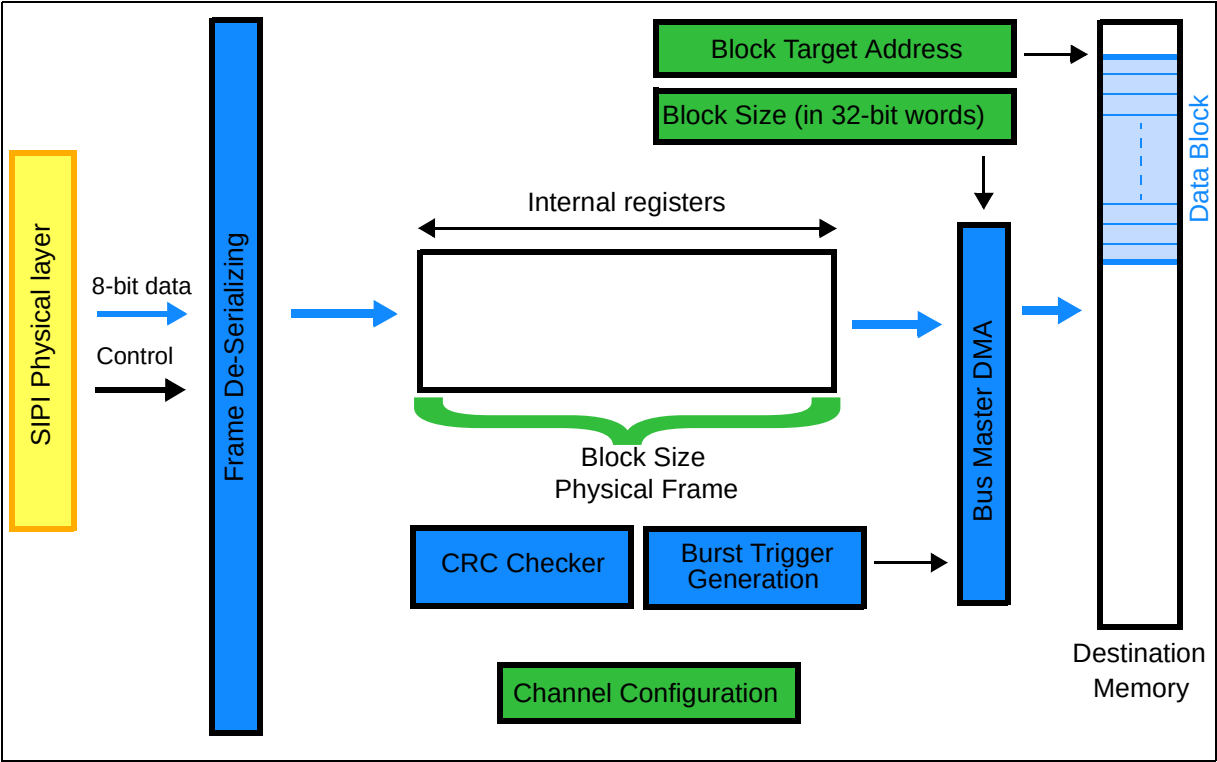
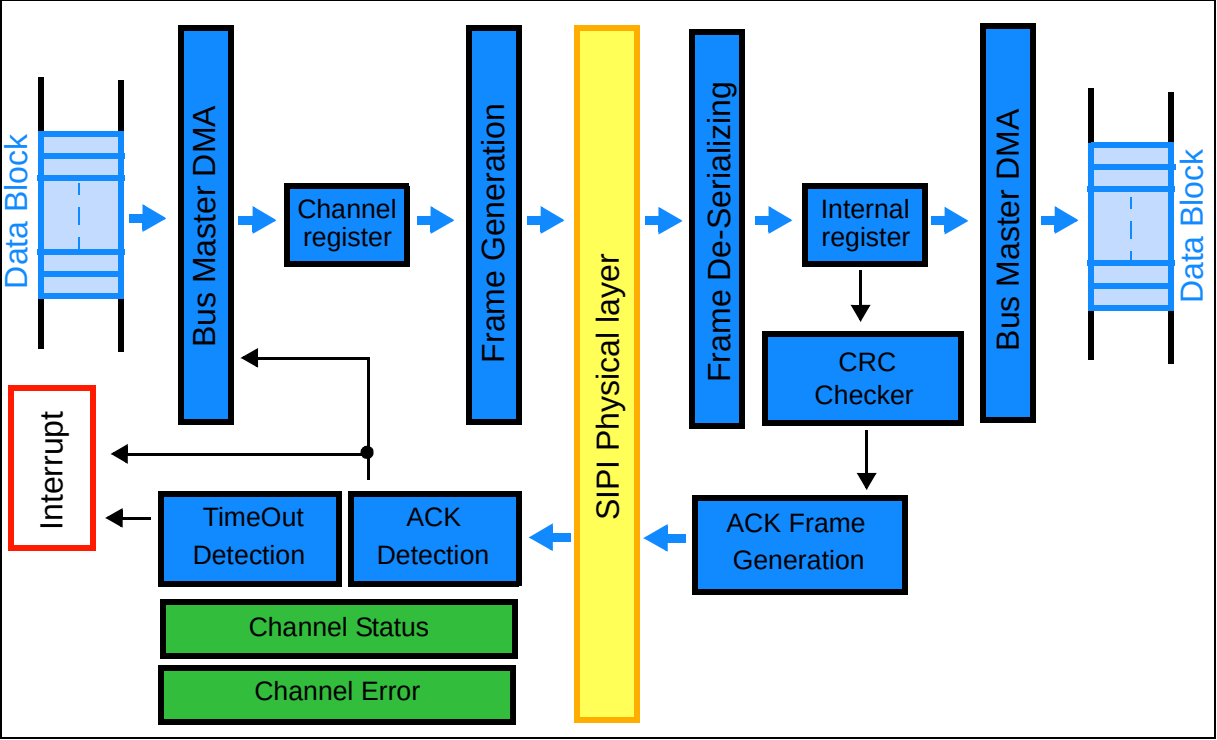


Figure 715. SIPI data stream with acknowledge trigger



47.8 DMA programming sequence

1. Software configures TCD (DMA data).
2. [BLG_SIPI_0004][Covers: ADD20.243]DMA transfers from RAM to SIPI registers.[end]
3. [BLG_SIPI_0005][Covers: ADD20.243]SIPI starts transferring the data through LFAST Tx ports.[end]
4. If an acknowledge or fault response is received, SIPI will repeat the process. If any error response is received, SIPI will move to an idle state. An interrupt will be flagged by SIPI, and software will take control of the SIPI.
5. If SIPI has received a read response, and error flags are not set on the LFAST Rx port, SIPI will assert a request. In case of an error, move to Step ⁽⁸⁾.
6. As soon as the request is asserted, DMA transfer starts between the SIPI registers and RAM.
7. If DMA transfer minor loop/major loop is complete, the request is negated.
8. If an error occurs:
 - The SIPI generates an interrupt
 - Software takes control of the SIPI
9. Steps ⁽⁵⁾–⁽⁸⁾ are repeated until the transfer is finished.

47.9 Modes of operation

SIPI has three operating modes: Initialization, Normal and Module Disable modes. After hardware reset the SIPI is in module disable mode, which helps reduce power consumption.

47.9.1 Initialization mode (INIT)

To enter INIT mode software writes SIPIMCR[INIT] = 1 (SIPIMCR[MOEN] = 1 must be written before attempting to write SIPIMCR[INIT]). To exit initialization mode software writes SIPIMCR[INIT] = 0.

When in initialization mode, all message transfers are stopped. If software invokes initialization mode when a bus transfer is in progress, the transfer will be aborted immediately whether the transfer has completed, or not.

Note: It is recommended that software checks the SIPI state (SIPIMCR[MOEN]) before setting SIPIMCR[INIT].

47.9.2 Normal mode (NM)

Once software has completed initialization of SIPI (SIPIMCR[INIT] = 1), it can enter Normal mode by writing SIPIMCR[INIT] = 0. SIPI needs to be in Normal mode for all data transfers.

Note: .SIPI must be enabled before attempting to write to SIPIMCR[INIT] (for example, SIPIMCR[MOEN] = 1).

47.9.3 Module disable (MD)

Module Disable mode in the SIPI is used to help reduce power consumption. By default, SIPI is in disable mode, SIPIMCR[MOEN] = 0, and is exited by writing SIPIMCR[MOEN] = 1. All the activities on the SIPI Tx and Rx ports are immediately stopped in Module Disable mode.

47.10 Errors

This section describes the potential errors that can occur during SIPI operation.

47.10.1 Timeout error

This error is generated at the initiator node when the acknowledge/response is not received within the configured time in the corresponding $CTORn[TOR]$ field setting. $SIPERR[TOEn] = 1$ indicates that a channel timeout error has occurred.

Note: SIPI should not drop the response even after timeout occurs. Software polls both error and status flags after the transfer to see if there was a timeout error. If there was a timeout error the response received may be discarded.

47.10.2 CRC error

A CRC error is generated at the target nodes when the CRC received with the frame does not match with the calculated CRC. When a CRC error is detected the $SIPISR[GCRCE]$ is set and an interrupt will be generated if $SIPIMCR[CRCIE] = 1$.

Note: In the case of a CRC error, the target node will not send an acknowledge to the initiator node. Then an interrupt will be generated on the target side if the corresponding interrupt enable bit is set. The initiator node will detect a time-out and take necessary action.

47.10.3 Maximum count reached error

This error is always generated at the target node only. The maximum count reached error is generated when the value of the $SIPACR$ equals $MAXCR$ (see [Section 47.14.1.12, SIPI Address Count Register \(SIPACR\)](#) and [Section 47.14.1.10, Max Count Register \(MAXCR\)](#) for more details). The $SIPISR[MCR]$ will be set, and an interrupt will be generated if $SIPIMCR[MCRIE] = 1$.

47.10.4 TID error

This error is always generated at the initiator node only. It is generated when header bits 15–13 do not match $CSRn[TID]$ (transaction ID bits), (see [Section 47.14.1.2, Channel Status Register \(CSRn\)](#)). The $CSRn[TIDE]$ bit will be set when a TID error is detected, and an interrupt will be generated if $CIRn[TIDIE] = 1$.

47.10.5 Acknowledge error

When an incorrect Acknowledge is received the $SIPERR[ACKRn]$ will be set, at the initiator node only, and an interrupt will be generated if $CIRn[WAIE] = 1$.

47.11 CRC calculation

Example: If header is AAB Bh, address is 1122_3344h and data is CCDD_EEFFh. Then CRC calculation will take place as follows:

- 1) The CRC seed is initialized by FFFF_FFFFh.
- 2) All the data will be mirrored before sending to CRC IP (for example, MSB will be sent as LSB).

3) So the header will be sent as DD55_0000h.

4) Address will be sent as 22CC_4488h.

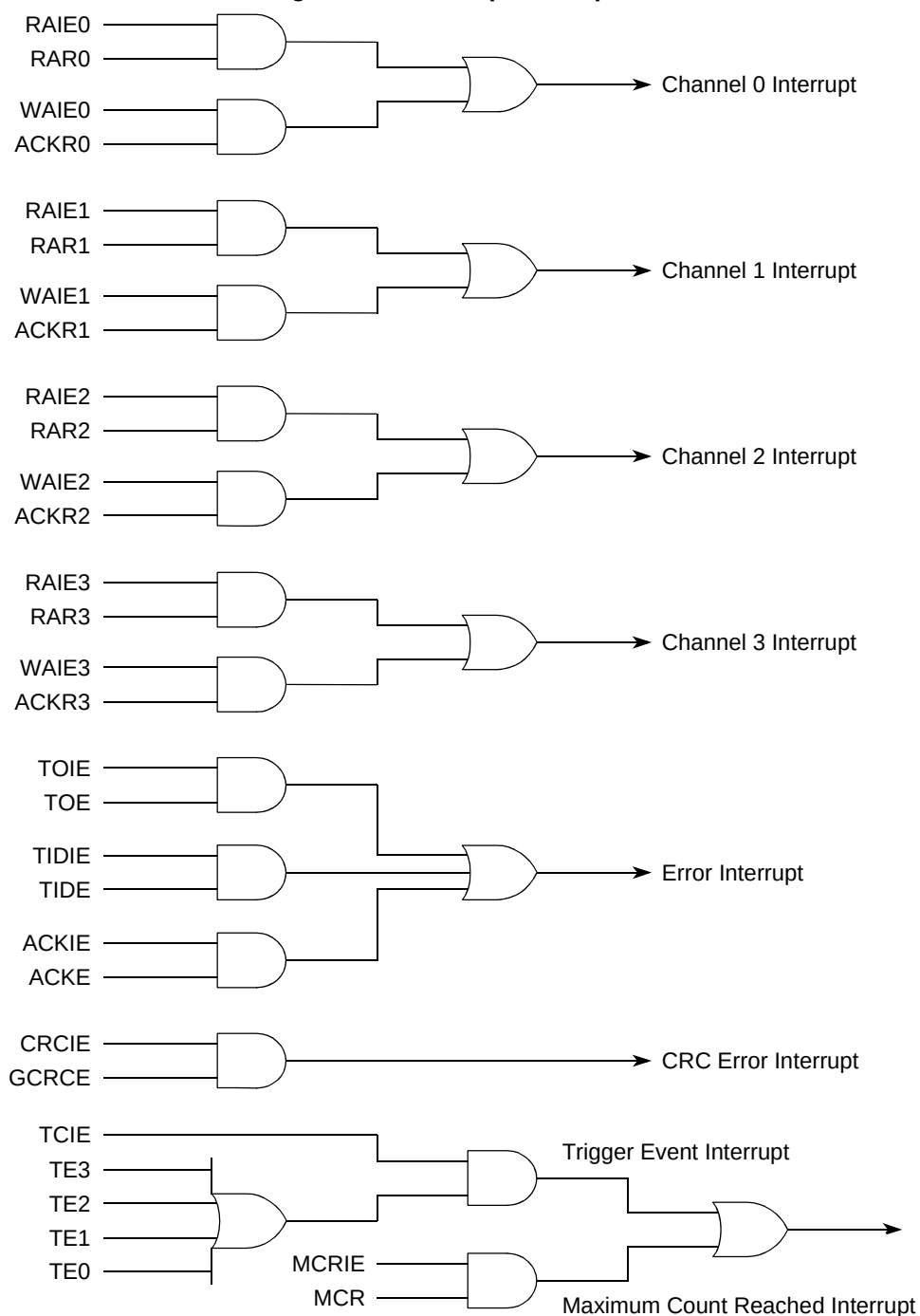
5) Data will be sent as FF77_BB33h.

Note: Please refer to the Cyclic Redundancy Check Unit (CRC) chapter in this manual for a detailed description of CRC generation and operation.

47.12 Interrupt logic

A description of the interrupt logic can be found in [Figure 716](#)

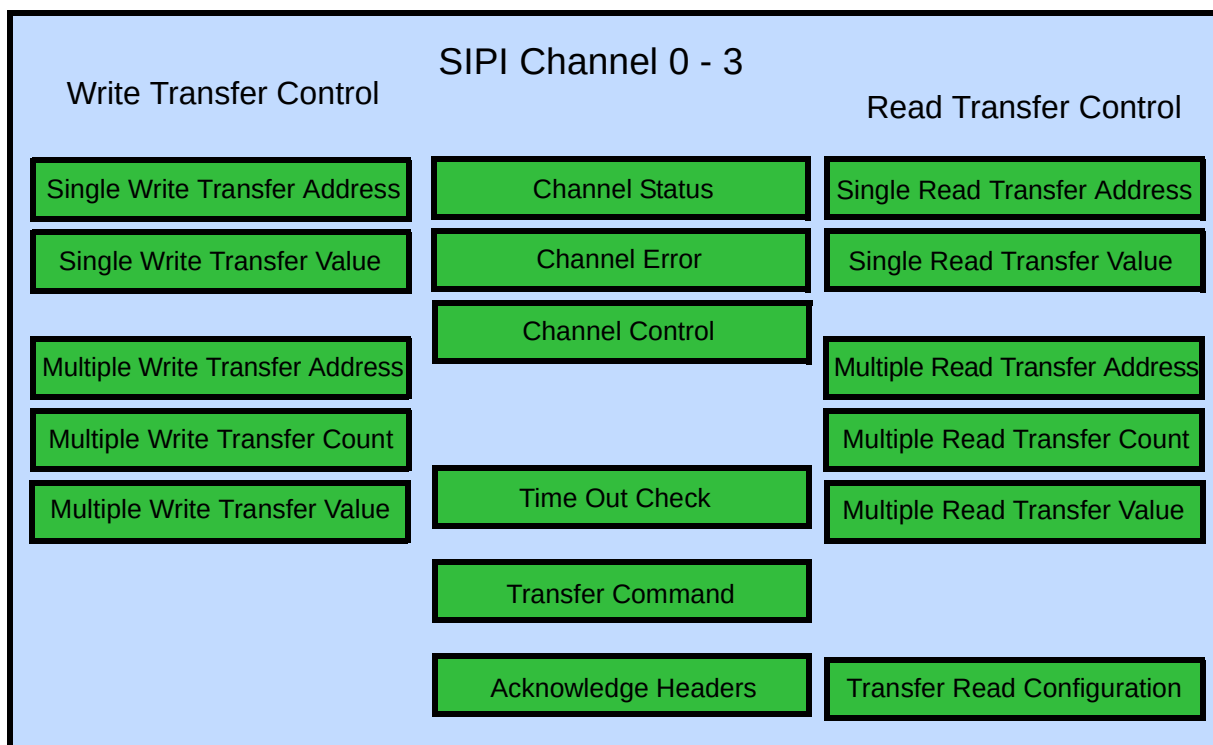
Figure 716. Interrupt description



47.13 SIPI control and status overview

The diagram below shows the relationship between transfers and the SIPI control and status registers.

Figure 717. SIPI control & status overview – Register transfer



47.14 Memory map and register description

Table 762. SIPI memory map

Offset from SIPI base (hex)	Register	Section/Page
0000	SIPI Channel Control Register 0 (CCR0)	47.14.1.1/1612
0004	SIPI Channel Status Register 0 (CSR0)	47.14.1.2/1613
0008	Reserved	—
000C	SIPI Channel Interrupt Register 0 (CIR0)	47.14.1.3/1614
0010	SIPI Channel Timeout Register 0 (CTOR0)	47.14.1.4/1615
0014	SIPI Channel CRC Register 0 (CCRC0)	47.14.1.5/1616
0018	SIPI Channel Address Register 0 (CAR0)	47.14.1.6/1617
001C	SIPI Channel Data Register 0 (CDR0)	47.14.1.7/1618
0020	SIPI Channel Control Register 1 (CCR1)	47.14.1.1/1612
0024	SIPI Channel Status Register 1 (CSR1)	47.14.1.2/1613
0028	Reserved	—
002C	SIPI Channel Interrupt Register 1 (CIR1)	47.14.1.3/1614

Table 762. SIPI memory map (continued)

Offset from SIPI base (hex)	Register	Section/Page
0030	SIPI Channel Timeout Register 1 (CTOR1)	47.14.1.4/1615
0034	SIPI Channel CRC Register 1 (CCRC1)	47.14.1.5/1616
0038	SIPI Channel Address Register 1 (CAR1)	47.14.1.6/1617
003C	SIPI Channel Data Register 1 (CDR1)	47.14.1.7/1618
0040	SIPI Channel Control Register 2 (CCR2)	47.14.1.1/1612
0044	SIPI Channel Status Register 2 (CSR2)	47.14.1.2/1613
0048	Reserved	—
004C	SIPI Channel Interrupt Register 2 (CIR2)	47.14.1.3/1614
0050	SIPI Channel Timeout Register 2 (CTOR2)	47.14.1.4/1615
0054	SIPI Channel CRC Register 2 (CCRC2)	47.14.1.5/1616
0058	SIPI Channel Address Register 2 (CAR2)	47.14.1.6/1617
005C	SIPI Channel Data Register 2[0] (CDR2[0])	47.14.1.7/1618
0060	SIPI Channel Data Register 2[1] (CDR2[1])	47.14.1.7/1618
0064	SIPI Channel Data Register 2[2] (CDR2[2])	47.14.1.7/1618
0068	SIPI Channel Data Register 2[3] (CDR2[3])	47.14.1.7/1618
006C	SIPI Channel Data Register 2[4] (CDR2[4])	47.14.1.7/1618
0070	SIPI Channel Data Register 2[5] (CDR2[5])	47.14.1.7/1618
0074	SIPI Channel Data Register 2[6] (CDR2[6])	47.14.1.7/1618
0078	SIPI Channel Data Register 2[7] (CDR2[7])	47.14.1.7/1618
007C	SIPI Channel Control Register 3 (CCR3)	47.14.1.1/1612
0080	SIPI Channel Status Register 3 (CSR3)	47.14.1.2/1613
0084	Reserved	—
0088	SIPI Channel Interrupt Register 3 (CIR3)	47.14.1.3/1614
008C	SIPI Channel Timeout Register 3 (CTOR3)	47.14.1.4/1615
0090	SIPI Channel CRC Register 3 (CCRC3)	47.14.1.5/1616
0094	SIPI Channel Address Register 3 (CAR3)	47.14.1.6/1617
0098	SIPI Channel Data Register 3 (CDR3)	47.14.1.7/1618
009C	SIPI Module Configuration Register (SIPIMCR)	47.14.1.8/1619
00A0	SIPI Status Register (SIPISR)	47.14.1.9/1621
00A4	SIPI Max Count Register (MAXCR)	47.14.1.10/1622
00A8	SIPI Address Reload Register (SIPIARR)	47.14.1.11/1623
00AC	SIPI Address Count Register (SIPIACR)	47.14.1.12/1623
00B0	SIPI Error Register (SIPIERR)	47.14.1.13/1624

47.14.1 Register descriptions

47.14.1.1 Channel Control Register (CCRn)

Note: Back to back transactions are not supported on a single channel (for example, if the channel busy bit, $CSRn[CB] = 1$, is configured for 1 channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, $CSRn[CB] = 0$).

PRIORITY SCHEDULING: The channel whose $CARn$ is written first will have access first, irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Priority scheduling occurs when the LFAST is unable to receive data, and no channel is pending transmission. The reason for the scheduling is that a common CRC module is shared by all channels.

Note: This register can only be written during Initialization mode ($MCR[INIT] = 1$), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted ($CSRn[CB] = 1$, [Section 47.14.1.2, Channel Status Register \(CSRn\)](#)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding $CARn$ ([Section 47.14.1.6, Channel Address Register \(CARn\)](#)).

Figure 718. Channel Control Register (CCRn)

0000h (CCR0)																					
Offset:	0020h (CCR1)																				Access: User read/write
	0040h (CCR2)																				
	007Ch (CCR3)																				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15					
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TC ⁽¹⁾				
W																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31					
R	0	0	0	0	0	0	0	0	WL[1:0]		CHEN	ST ⁽²⁾	IDT ⁽¹⁾	RRT ⁽¹⁾	WRT ^{(1),(2)}	DEN					
W																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

1. The order of priority for writing to IDT, WRT and RRT must be in this order: 1) IDT, 2) RRT, 3) WRT.
2. Only transfers with channel 2 can write 1 to this bit. All other channels forces $CCRn[ST] = 0$. The $CCR2[WRT]$ and $CCR2[ST]$ bits must be set for streaming.

Table 763. CCR*n* field descriptions

Field	Description
0–14	Reserved
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in the CAR. 0 Trigger command not sent 1 Trigger command sent
16–23	Reserved
24–25 WL[1:0]	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all the channel enables are enabled simultaneously, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0 for channels 0, 1 and 3. It can only be written for channel 2 (CCR2). This bit can only be written when CCR <i>n</i> [WRT] = 1. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. 0 Read request will not be sent 1 Read request transfer by the initiator. This bit cannot be written if CCR <i>n</i> [IDT] = 1.
30 WRT	Write Request Transfer. 0 No write request will be sent 1 Write request transfer by the initiator. This bit cannot be written if CCR <i>n</i> [IDT] = 1 or CCR <i>n</i> [RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after 1 major loop has completed. 0 Channel will be used for bus interface access 1 Channel will be used for DMA access

47.14.1.2 Channel Status Register (CSR*n*)

The CSR*n* contains the status bits for the current transfer.

Figure 719. Channel Status Register (CSR_n)

0004h (CSR0)
Offset: 0024h (CSR1)
0044h (CSR2)
0080h (CSR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RAR	TID[2:0]			ACKR	CB	0	0
W									w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 764. CSR_n field descriptions

Field	Description
0–23	Reserved
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID[2:0]	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel status. This bit is read only. 0 Channel free 1 Channel busy
30–31	Reserved

47.14.1.3 Channel Interrupt Register (CIR_n)

The CIR_n contains the interrupt enable bits for the corresponding channel (*n*).

Figure 720. Channel Interrupt Register (CIR_n)

000Ch (CIR0)
Offset: 002Ch (CIR1)
004Ch (CIR2)
0088h (CIR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0						
W											WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 765. CIR_n field descriptions

Field	Description
0–25	Reserved
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

47.14.1.4 Channel Timeout Register (CTOR_n)

The CTOR_n contains the timeout value for Tx requests.

Figure 721. Channel Timeout Register (CTORn)

0010h (CTOR0)
Offset: 0030h (CTOR1)
0050h (CTOR2)
008Ch (CTOR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	TOR ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

1. Field can only be written during Initialization mode (MCR[INIT] = 1).

Table 766. CTORn field descriptions

Field	Description
0–23	Reserved
24–31 TOR	Timeout value for transmitted requests. These bits can only be written by software in initialization mode, SIPIMCR[INIT] = 1. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPIMCR[PRSCLR].

47.14.1.5 Channel CRC Register (CCRCn)

CCRCn is read-only register that returns the CRC calculated value on the header, address, and data bits.



Figure 722. Channel CRC Register (CCRCn)

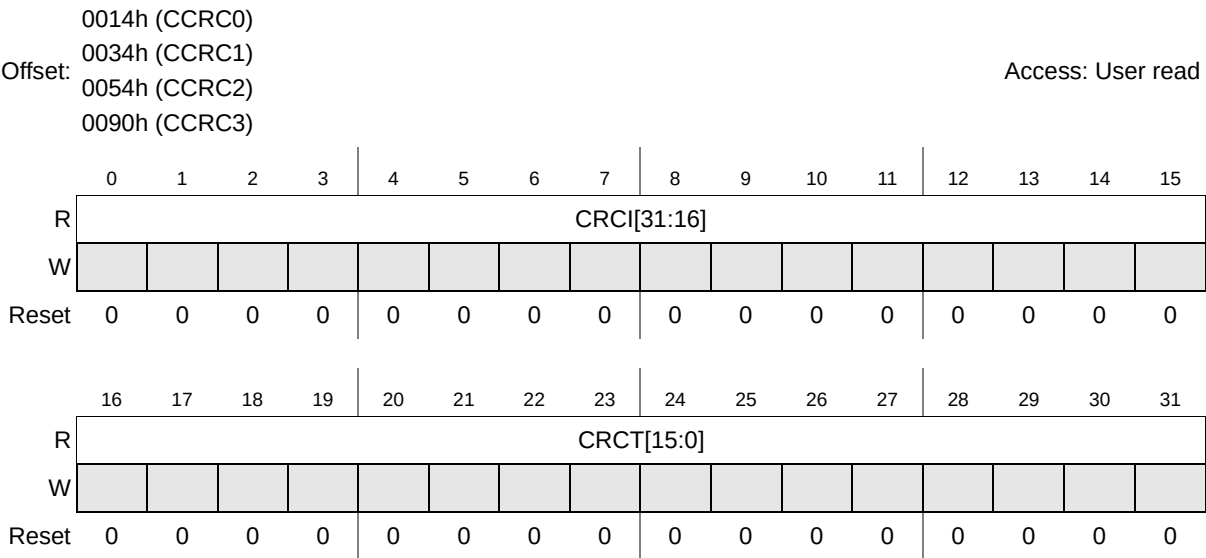


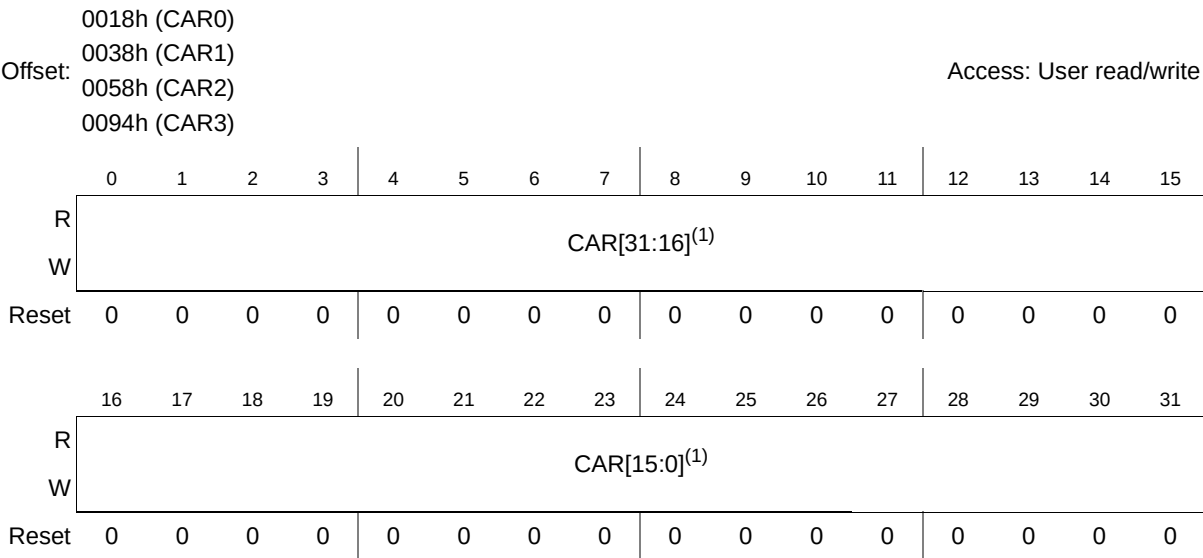
Table 767. CCRCn field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

47.14.1.6 Channel Address Register (CARn)

The CAR contains the data transmission target address. For streaming operations this register contains the start address. The CARn is a 32-bit register and can be written anytime by software or DMA.

Figure 723. Channel address Register (CAR_n)



1. This register holds the start address of streaming operations.

Table 768. CAR_n field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

47.14.1.7 Channel Data Register (CDR_n)

The CDR_n contains the data to be transmitted, and can be written anytime by software or DMA. The data can be written in 8, 16 or 32 bit formats.

Figure 724. Channel Data Register (CDR_n)

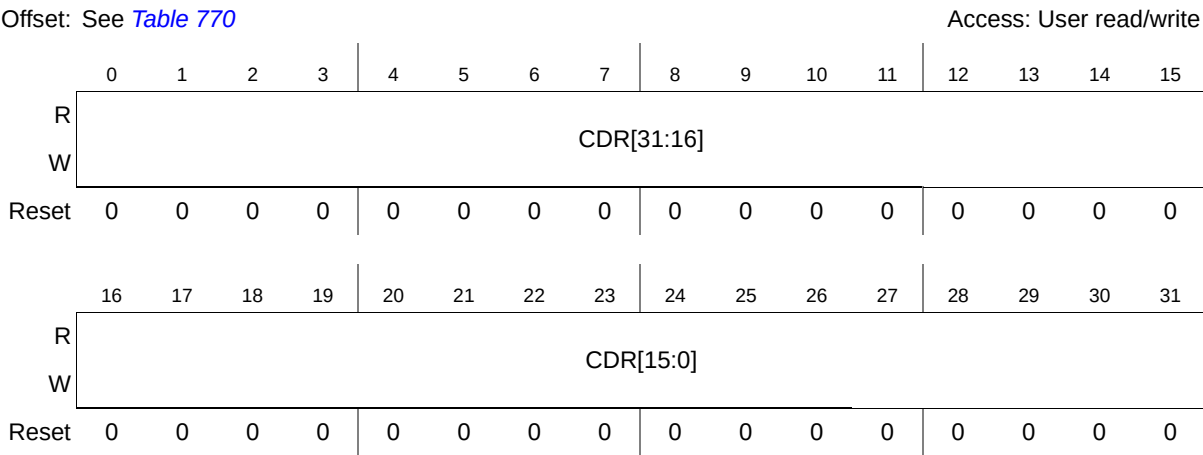


Table 769. CDR_n field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

Table 770. Channel data Register address offsets

Register	Address Offset (hex)
CDR0	001Ch
CDR1	003Ch
CDR2[0]–CDR2[7]	005Ch, 0060h, 0064h, 0068h, 006Ch, 0070h, 0074h, 0078h
CDR3	0098h

47.14.1.8 SIPI Module Configuration Register (SIPIMCR)

The SIPIMCR is a 32-bit global configuration register of the SIPI.

Figure 725. SIPI Module Configuration Register (SIPIMCR)

Offset: 009Ch												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZ		HALT	0	0	PRSCLR ⁽¹⁾										
W		0														
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AID ⁽¹⁾		0	0	0	CRCIE ⁽²⁾	MCRIE ⁽²⁾	0	0	0	0	CHNSB ³	TEN ⁽²⁾	INIT ⁽²⁾	MOEN	SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read/cleared by software. Can be written in initialization mode only (SIPIMCR[INIT] = 1).

2. Can only be written when SIPIMCR[MOEN] = 1.

3. Can be written only once after reset.

Table 771. SIPIMCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
1	Reserved
2 HALT	Halt Mode Enable. Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. 0 No Freeze mode request 1 Enters Freeze mode if FRZ bit is asserted.
3–4	Reserved
5–15 PRSCLR	These bits should be programmed by software in initialization mode, SIPIMCR[INIT] = 1, writes during other times are ignored. The timeout counter runs on the prescaled system clock. Default value is 64 (040h). The allowed programmable values are: 040h 64 080h 128 100h 256 200h 512 400h 1024
16–17 AID	Address Increment/Decrement. These bits define the type of address change at the target node. 00 no change. address stays same 01 address increments by 4 10 address decrements by 4. 11 Reserved
18–20	Reserved
21 CRCIE	CRC Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
22 MCRIE	Max Count Reached Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
23–26	Reserved
27 CHNSB	Channel coding select bit 0, 1 : Channel number select as per Table 759, SIPI Header Channel Number Coding .
28 TEN	Target Enable. Setting this bit enables the target functionality at SIPI. This bit can be read or written anytime. This bit is automatically negated by hardware when target detects an error in “streaming without ACK” mode. This bit has to be enabled for transmission operations as well.
29 INIT	Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPIMCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits cannot be enabled together. 0 Normal Mode 1 Initialization Mode

Table 771. SIPIMCR field descriptions (continued)

Field	Description
30 MOEN	Module Enable. This bit should be set or cleared by software. When this bit is negated, all SIPI operations are immediately stopped.
31 SR	Soft Reset. Setting this bit clears all status and error registers and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete.

47.14.1.9 SIPI Status Register (SIPI SR)

The SIPI SR is the global status register of SIPI.

Figure 726. SIPI Status Register (SIPI SR)

Offset: 00A0h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZACK	LPMACK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	GCRCE	MCR	0	TE[3:0]				STATE			
W						w1c	w1c		w1c	w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 772. SIPI SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge. This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited. 0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge This read-only bit indicates that SIPI is in Disable Mode. Either of these low power modes cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI not in any low power modes 1 SIPI is either in Disable Mode
2–20	Reserved

Table 772. SIPI SR field descriptions (continued)

Field	Description
21 GCRCE	Global CRC Error Bit 0 No CRC error 1 CRC error occurred
22 MCR	Maximum Count Reached. This bit will be set whenever SIPIACR[ADCNT] = MAXCR[MXCNT]. An interrupt will be generated when this bit is set only if SIPIMCR[MCRIE] = 1. This bit should be cleared by software.
23	Reserved
24–27 TE	Trigger Event on Respective Channels. This field enables interrupts for target nodes. TE3 = 1 - Channel 3 trigger event TE2 = 1 - Channel 2 trigger event TE1 = 1 - Channel 1 trigger event TE0 = 1 - Channel 0 trigger event
28–31 STATE	These bits reflect the transmit state machine status. They can be polled for determination of state machine status. 0000 IDLE 0001 HEADER_AND_ADDRESS_FIELD 0010 HEADER_AND_DATA_FIELD 0011 HEADER_AND_CRC_FIELD 0100 ADDRESS_AND_CRC_FIELD 0101 ADDRESS_AND_DATA_FIELD 0110 DATA_AND_CRC_FIELD 0111 DATA_FIELD

47.14.1.10 Max Count Register (MAXCR)

The MAXCR contains the maximum address count value at target node. It is programmed via direct write request through the initiator. This register can be read or written by software anytime.

Figure 727. Max Count Register (MAXCR)

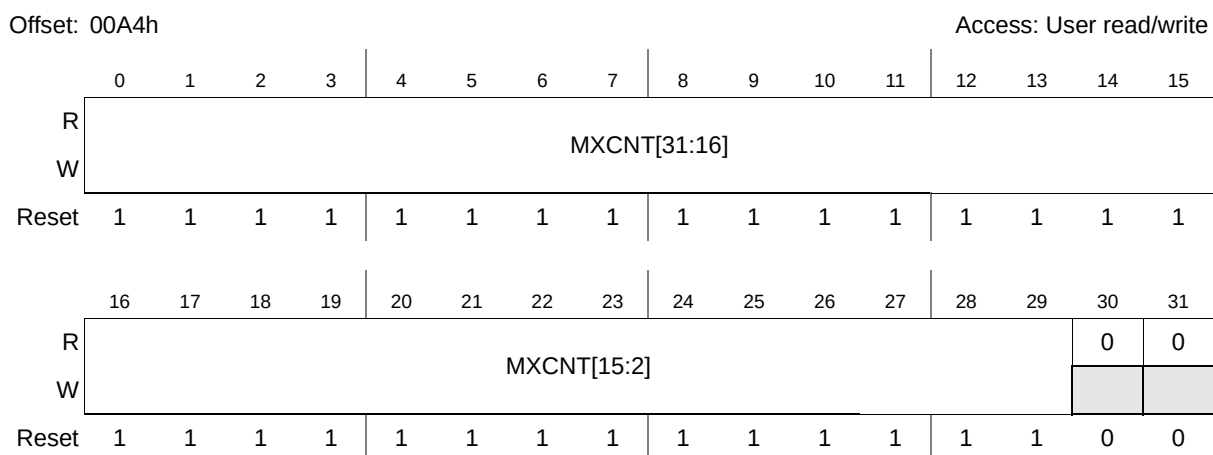


Table 773. MAXCR field descriptions

Field	Description
0–29 MXCNT	This register contains the maximum address count value at target node. It should be programmed via direct write request through initiator.
30–31	Reserved

47.14.1.11 SIPI Address Reload Register (SIPIARR)

The SIPIARR contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator. This register can be read or written anytime.

Figure 728. SIPI Address Reload Register (SIPIARR)

Offset: 00A8h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADRLD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADRLD[15:2]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 774. SIPIARR field descriptions

Field	Description
0–29 ADRLD	Contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.
30–31	Reserved

47.14.1.12 SIPI Address Count Register (SIPIACR)

This register reflects the count value of address counter at target node. It should be configured by direct write request from initiator. This register can be read/written by software anytime.

Figure 729. SIPI Address Count Register (SIPIACR)

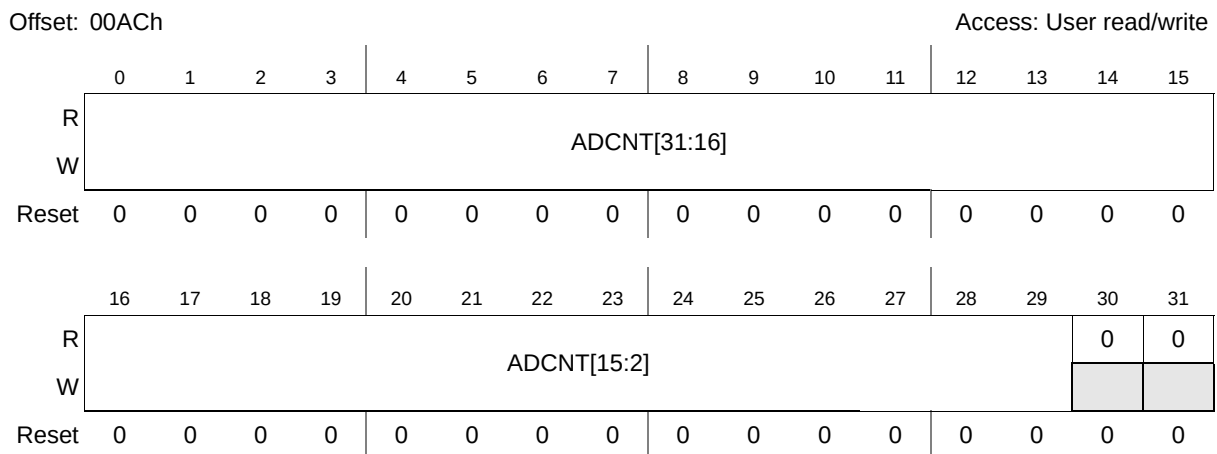


Table 775. SIPIACR field descriptions

Field	Description
0–29 ADCNT	Reflects the count value of address counter at target node. It should be configured by direct write request from the initiator. This register can be read/written by software anytime. At the end of the streaming operation, SIPIACR will point to the next address to be written.
30-31	Reserved

Note: *SIPIARR, SIPIACR and MAXCR will be configured by direct write operation through the initiator.*

[BLG_SIP_0003][Covers: ADD20.244] When a streaming write command is received, the target will start the write operation from the address present in SIPIACR. [end]

[BLG_SIP_0007][Covers: ADD20.244] After each 32-bit write has completed, SIPIACR[ADCNT] is compared against MAXCR[MXCNT]. If they are equal, SIPIACR[ADCNT] is loaded with the value stored in SIPIARR[ADRLD]. If they are not equal, the SIPIACR[ADCNT] will increment by 4, decrement by 4, or remain the same (depending on configuration). In either case, the SIPIACR[ADCNT] will contain the address at which the next streaming write will occur. [end]

47.14.1.13 SIPI Channel Error Register (SIPIERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate communication errors.

Figure 730. SIPI Channel Error Register (SIPIERR)

Offset: 00B0h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	TOE3	TIDE3	ACKE3	0	0	0	0	0	TOE2	TIDE2	ACKE2
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	TOE1	TIDE1	ACKE1	0	0	0	0	0	TOE0	TIDE0	ACKE0
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 776. SIPIERR field descriptions

Field	Description
0–4	Reserved
5 TOE3	Timeout Error for Channel 3. 0 Timeout error occurred 1 Timeout error did not occur
6 TIDE3	Transaction ID Error for Channel 3. 0 Received transaction ID matched with the stored ID 1 Received transaction ID did not match with the stored ID
7 ACKE3	Acknowledge Error for Channel 3. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
8–12	Reserved
13 TOE2	Timeout Error for Channel 2. 0 Timeout error occurred 1 Timeout error did not occur
14 TIDE2	Transaction ID Error for Channel 2. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
15 ACKE2	Acknowledge Error for Channel 2. 0 Acknowledge received is correct 1 Acknowledge received is not correct
16–20	Reserved
21 TOE1	Timeout Error for Channel 1. 0 Timeout error occurred 1 Timeout error didn't occur

Table 776. SIPIERR field descriptions (continued)

Field	Description
22 TIDE1	Transaction ID Error for Channel 1. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
23 ACKE1	Acknowledge Error for Channel 1. 0 Acknowledge received is correct 1 Acknowledge received is not correct
24–28	Reserved
29 TOE0	Timeout Error for Channel 0. 0 Timeout error occurred 1 Timeout error didn't occur
30 TIDE0	Transaction ID Error for Channel 0. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
31 ACKE0	Acknowledge Error for Channel 0. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

48 Deserial Serial Peripheral Interface (DSPI)

48.1 Introduction

The Deserial Serial Peripheral Interface (DSPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device. The module supports MCU pin count reduction through serialization and deserialization of MCU internal signals transmitted over the SPI serial link.

Note: For the chip-specific implementation details of this module's instances, see the *Device Configuration* chapter.

48.1.1 Block diagram

Figure 731 provides the block diagram of this module.

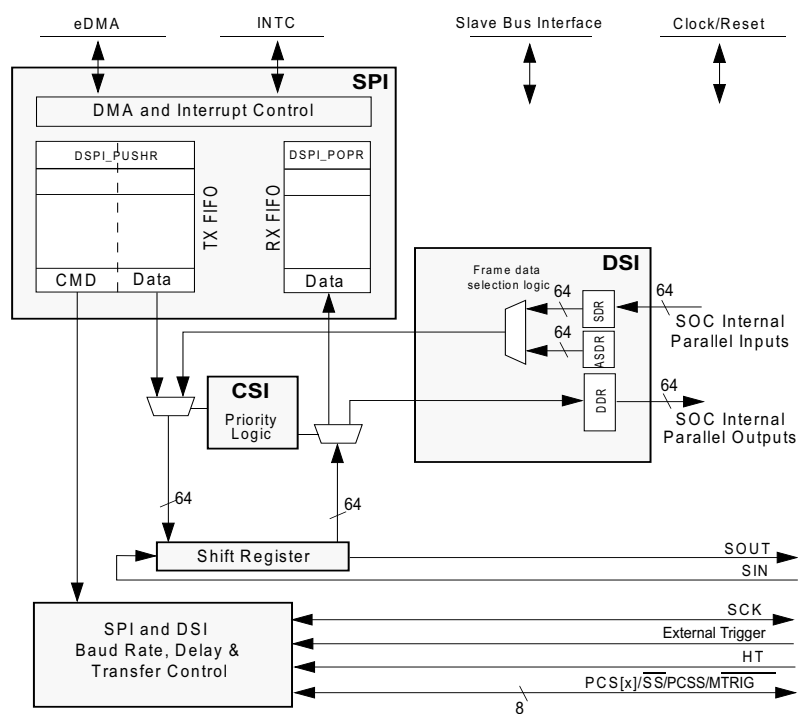


Figure 731. DSPI block diagram

48.1.2 Features

The DSPI supports these features:

- full-duplex, four-wire synchronous transfers
- Master and Slave modes: data streaming operation in slave mode with continuous slave selection
- buffered transmit operation using the TX FIFO with depth of 4 entries
- buffered receive operation using the RX FIFO with depth of 4 entries
- asynchronous clocking scheme for Register and Protocol interfaces
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- visibility into TX and RX FIFOs for ease of debugging
- programmable transfer attributes on a per-frame basis:
 - 8 transfer attribute registers along with 8 extended transfer attribute registers
 - serial clock with programmable polarity and phase
 - various programmable delays
 - programmable serial frame size of 4 to 64 bits, expandable by software control: SPI frames longer than 32 bits can be supported using the continuous selection format
 - continuously held chip select capability
 - parity control
- 8 Peripheral Chip Selects, expandable to 256 with external demultiplexer
- deglitching support for up to 128 Peripheral Chip Selects with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
 - TX FIFO is not full (TFFF)
 - RX FIFO is not empty (RFDF)
 - CMD FIFO is not full (CMDFFF)
- interrupt conditions:
 - End Of Queue reached (EOQF)
 - TX FIFO is not full (TFFF)
 - CMD FIFO is not full (CMDFFF)
 - Transfer of current frame Complete (TCF)
 - Transfers due from current Command frame Complete (CMDTCF)
 - Transfer of current SPI frame Complete (SPITCF)
 - attempt to transmit with an empty Transmit FIFO (TFUF)
 - RX FIFO is not empty (RFDF)
 - frame received while Receive FIFO is full (RFOF)
 - SPI Parity Error (SPEF)
 - data present in TX FIFO while CMD FIFO is empty (TFIWF)
- global interrupt request line
- modified SPI transfer formats for communication with slower peripheral devices
- power-saving architectural features
 - support for stop mode
 - support for doze mode

The DSPI also supports pin reduction through serialization and deserialization if enabled for the module.

- two sources of serialized data:
 - DSPI memory-mapped register
 - parallel input signals
 - programmable selection of source data on bit basis
- deserialized data provided as Parallel Output signals and as bits in a memory-mapped register
- interrupt conditions:
 - deserialized data matches pre-programmed pattern (DDIF)
 - Transfer of current DSI frame Complete (DSITCF)
 - DSI Parity Error (DPEF)
- DMA request support for following conditions: deserialized data matches pre-programmed pattern (DDIF)
- transfer initiation conditions:
 - continuous
 - edge sensitive hardware trigger
 - change in data
- support for parallel and serial chaining of several DSPI modules inside the MCU
- pin serialization/deserialization with interleaved SPI frames for control and diagnostics
- continuous serial communications clock

DSPI supports a combination of SPI and DSI modes of operation (Combined Serial Interface (CSI)) for the downstream Micro Second Channel in either Timed Serial Bus (TSB) configuration or Interleaved Frames Configuration (configurable by software). Both TSB and Interleaved TSB modes have the following common features:

- transmission of frames is done at frame boundaries
- frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame
- separate interrupts for frame completion from SPI and DSI

48.1.3 DSPI configurations

The DSPI module can operate in three configurations: SPI, DSI, and CSI.

48.1.3.1 SPI configuration

The SPI configuration allows the DSPI to send and receive serial data. This configuration allows the DSPI to operate as a basic SPI block with internal FIFOs supporting external queues operation. Transmit data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the receive FIFO and write transmit data to the transmit FIFO.

For queued operations, the SPI queues can reside in system RAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished by a DMA controller or host CPU. [Figure 732](#) shows a system example with DMA, DSPI and external queues in system RAM.

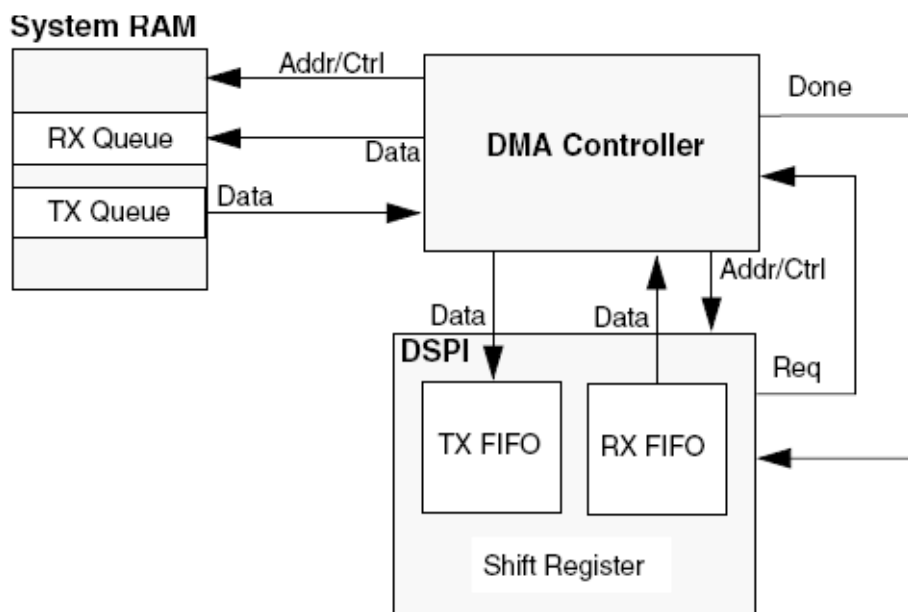


Figure 732. DSPI with queues and DMA

48.1.3.2 DSI configuration

In the DSI configuration the DSPI serializes up to 64 parallel input signals or register bits. The DSPI also deserializes the received data to parallel output signals or to a memory-mapped register. The data is transferred using a SPI-like protocol.

TSB mode, detailed on [Section 48.4.9, Timed Serial Bus \(TSB\)](#), provides the Micro Second downstream Channel support (MSC), serializing from 4 to 64 parallel input signals or register bits. [Section 46.4.9, Timed Serial Bus \(TSB\)](#)

48.1.3.3 CSI configuration

There are three configurations available in this mode.

- The normal CSI configuration is a combination of the SPI and DSI configurations. In this configuration the DSPI interleaves DSI data frames with SPI data frames. Interleaving is done on the frame boundaries.
- In the TSB configuration, transmission of SPI data has higher priority than DSI data.
- In the Interleaved TSB (ITSB) configuration, the frames from SPI and DSI are interleaved without priority. On every trigger, frames from DSI are sent when there are no frames in the SPI or the previous transmission was a frame from SPI. ITSB is detailed in [Section 48.4.10, Interleaved TSB \(ITSB\) Mode](#).

48.1.4 Modes of operation

The DSPI modes of operation that can be divided into two categories:

- Module-specific modes
 - Master mode
 - Slave mode
 - Module Disable mode
- MCU-specific modes
 - External Stop mode
 - Debug mode

The DSPI enters module-specific modes when the host writes a DSPI register. The MCU-specific modes are controlled by signals external to the DSPI. The MCU-specific modes are modes that an MCU may enter in parallel to the DSPI block-specific modes.

48.1.4.1 Master mode

Master mode allows the DSPI to initiate and control serial communication. In this mode, the SCK signal and the PCS[x] signals are controlled by the DSPI and configured as outputs.

48.1.4.2 Slave mode

Slave mode allows the DSPI to communicate with SPI/DSI bus masters. In this mode, the DSPI responds to externally controlled serial transfers. The SCK signal and the PCS[0]/SS signals are configured as inputs and driven by a SPI bus master.

48.1.4.3 Module Disable mode

The Module Disable mode can be used for MCU power management. The clock to the non-memory-mapped logic in the DSPI can be stopped while in Module Disable mode.

48.1.4.4 External Stop mode

External Stop mode is used for MCU power management. The DSPI supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, the DSPI block acknowledges the request and completes the transfer in progress. When the DSPI reaches the frame boundary it signals that the protocol clock to the DSPI module may be shut off.

48.1.4.5 Debug mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls DSPI behavior in the debug mode. If the bit is set, the DSPI stops all serial transfers, when the MCU is in Debug mode. If the bit is cleared, the MCU Debug mode has no effect on the DSPI.

48.2 DSPI signal description

This section provides the DSPI signals description.

[Table 777](#) lists the signals that may connect off chip depending on device implementation.

Table 777. DSPI signal description

Signal	Description	I/O
PCS0/ SS	Master mode: Peripheral Chip Select 0 output Slave mode: Slave Select input	I/O
PCS[3:1]	Master mode: Peripheral Chip Select 1–3 Slave mode: Unused	O
PCS4/ MTRIG	Master mode: Peripheral Chip Select 4 Slave mode: Master Trigger	O
PCS5/ PCSS	Master mode: Peripheral Chip Select 5 / Peripheral Chip Select Strobe Slave mode: Unused	O
PCS[7:6]	Master mode: Peripheral Chip Select 6–7 Slave mode: Unused	O
SIN	Serial Data In	I
SOUT	Serial Data Out	O
SCK	Master mode: Serial Clock (output) Slave mode: Serial Clock (input)	I/O
HT	Hardware Trigger	I

48.2.1 $\overline{\text{PCS0/SS}}$ — Peripheral Chip Select/Slave Select

In master mode, the PCS0 signal is a Peripheral Chip Select output that selects which slave device the current transmission is intended for.

In slave mode, the active low $\overline{\text{SS}}$ signal is a Slave Select input signal that allows a SPI master to select the DSPI as the target for transmission.

48.2.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3

PCS1 – PCS3 are Peripheral Chip Select output signals in master mode.

In slave mode, these signals are unused.

48.2.3 $\overline{\text{PCS4/MTRIG}}$ — Peripheral Chip Select 4/Master Trigger

In master mode, PCS4 is a Peripheral Chip Select output signal.

In slave mode, the active low $\overline{\text{MTRIG}}$ is an output trigger signal that indicates that a change in data to be serialized has occurred. The MTRIG provides a pulse in DSI Configuration when a change in data to be serialized occurs. The MTRIG pulse is four protocol clock cycles in duration. If the DSPI is in slave mode and the MTO is disabled, the PCS4/ $\overline{\text{MTRIG}}$ signal is unused.

48.2.4 $\overline{\text{PCS5/PCSS}}$ — Peripheral Chip Select 5/Peripheral Chip Select Strobe

PCS5 is a Peripheral Chip Select output signal. When the DSPI is in master mode and the MCR[PCSSE] bit is cleared, this signal selects which slave device the current transfer is intended for.

When the DSPI is in master mode and the MCR[PCSSE] bit is set, the $\overline{\text{PCSS}}$ signal acts as a strobe to an external peripheral chip select demultiplexer, which decodes the PCS0 – PCS4 and PCS6 – PCS7 signals, preventing glitches on the demultiplexer outputs.

This signal is not used in slave mode.

48.2.5 PCS[6] – PCS[7] — Peripheral Chip Selects 6 – 7

PCS[6] – PCS[7] are Peripheral Chip Select output signals in master mode. In slave mode, these signals are not used.

48.2.6 SIN — Serial Input

SIN is a serial data input signal.

48.2.7 SOUT — Serial Output

SOUT is a serial data output signal.

48.2.8 SCK — Serial Clock

SCK is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK is an input from an external bus master.

48.2.9 HT — Hardware Trigger

HT is a trigger input signal that is used with Multiple Transfer Operations in DSI Configuration.

In master mode while in DSI or CSI Configurations, the HT signal initiates a data transfer when the DSICR0[TRRE] bit is set and a rising or falling edge is detected on HT. Which edge to trigger on is determined by the DSICR0[TPOL] bit.

In slave mode, the DSPI generates a trigger pulse on the $\overline{\text{MTRIG}}$ pin, when a rising or falling edge is detected on HT. Which edge that generates an output pulse is selected by the DSICR0[TPOL] bit.

48.2.10 Memory map/register definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the DSPIx_POPR also results in a transfer error.

Table 778. DSPI memory map

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_0000	DSPI Module Configuration Register (DSPIx_MCR)	32	R/W	0000_4001h	on page 1637
0000_0008	DSPI Transfer Count Register (DSPIx_TCR)	32	R/W	0000_0000h	on page 1640
0000_000C	DSPI Clock and Transfer Attributes Register 0 (In Master Mode) (DSPIx_CTAR0)	32	R/W	7800_0000h	on page 1641
0000_000C	DSPI Clock and Transfer Attributes Register 0 (In Slave Mode) (DSPIx_CTAR0_SLAVE)	32	R/W	7800_0000h	on page 1646

Table 778. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_0010	DSPI Clock and Transfer Attributes Register 1 (In Master Mode) (DSPIx_CTAR1)	32	R/W	7800_0000h	on page 1641
0000_0010	DSPI Clock and Transfer Attributes Register 1 (In Slave Mode) (DSPIx_CTAR1_SLAVE)	32	R/W	7800_0000h	on page 1646
0000_0014	DSPI Clock and Transfer Attributes Register 2 (In Master Mode) (DSPIx_CTAR2)	32	R/W	7800_0000h	on page 1641
0000_0018	DSPI Clock and Transfer Attributes Register 3 (In Master Mode) (DSPIx_CTAR3)	32	R/W	7800_0000h	on page 1641
0000_001C	DSPI Clock and Transfer Attributes Register 4 (In Master Mode) (DSPIx_CTAR4)	32	R/W	7800_0000h	on page 1641
0000_0020	DSPI Clock and Transfer Attributes Register 5 (In Master Mode) (DSPIx_CTAR5)	32	R/W	7800_0000h	on page 1641
0000_0024	DSPI Clock and Transfer Attributes Register 6 (In Master Mode) (DSPIx_CTAR6)	32	R/W	7800_0000h	on page 1641
0000_0028	DSPI Clock and Transfer Attributes Register 7 (In Master Mode) (DSPIx_CTAR7)	32	R/W	7800_0000h	on page 1641
0000_002C	DSPI Status Register (DSPIx_SR)	32	R/W	See register description	on page 1648
0000_0030	DSPI DMA/Interrupt Request Select and Enable Register (DSPIx_RSER)	32	R/W	0000_0000h	on page 1651
0000_0034	DSPI PUSH FIFO Register In Master Mode (DSPIx_PUSHR)	32	R/W	0000_0000h	on page 1654
0000_0034	DSPI PUSH FIFO Register In Slave Mode (DSPIx_PUSHR_SLAVE)	32	R/W	0000_0000h	on page 1657
0000_0038	DSPI POP FIFO Register (DSPIx_POPR)	32	R	0000_0000h	on page 1658
0000_003C	DSPI Transmit FIFO Register 0 (DSPIx_TXFR0)	32	R	0000_0000h	on page 1659
0000_0040	DSPI Transmit FIFO Register 1 (DSPIx_TXFR1)	32	R	0000_0000h	on page 1659
0000_0044	DSPI Transmit FIFO Register 2 (DSPIx_TXFR2)	32	R	0000_0000h	on page 1659
0000_0048	DSPI Transmit FIFO Register 3 (DSPIx_TXFR3)	32	R	0000_0000h	on page 1659
0000_007C	DSPI Receive FIFO Register 0 (DSPIx_RXFR0)	32	R	0000_0000h	on page 1660
0000_0080	DSPI Receive FIFO Register 1 (DSPIx_RXFR1)	32	R	0000_0000h	on page 1660
0000_0084	DSPI Receive FIFO Register 2 (DSPIx_RXFR2)	32	R	0000_0000h	on page 1660
0000_0088	DSPI Receive FIFO Register 3 (DSPIx_RXFR3)	32	R	0000_0000h	on page 1660

Table 778. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_00BC	DSPI DSI Configuration Register 0 (DSPIx_DSICR0)	32	R/W	0000_0000h	on page 1661
0000_00C0	DSPI DSI Serialization Data Register 0 (DSPIx_SDR0)	32	R	0000_0000h	on page 1664
0000_00C4	DSPI DSI Alternate Serialization Data Register 0 (DSPIx_ASDR0)	32	R/W	0000_0000h	on page 1665
0000_00C8	DSPI DSI Transmit Comparison Register 0 (DSPIx_COMPR0)	32	R	0000_0000h	on page 1666
0000_00CC	DSPI DSI Deserialization Data Register 0 (DSPIx_DDR0)	32	R	0000_0000h	on page 1667
0000_00D0	DSPI DSI Configuration Register 1 (DSPIx_DSICR1)	32	R/W	0000_0000h	on page 1668
0000_00D4	DSPI DSI Serialization Source Select Register 0 (DSPIx_SSR0)	32	R/W	0000_0000h	on page 1670
0000_00D8	DSPI DSI Parallel Input Select Register 0 (DSPIx_PISR0)	32	R/W	0000_0000h	on page 1671
0000_00DC	DSPI DSI Parallel Input Select Register 1 (DSPIx_PISR1)	32	R/W	0000_0000h	on page 1673
0000_00E0	DSPI DSI Parallel Input Select Register 2 (DSPIx_PISR2)	32	R/W	0000_0000h	on page 1674
0000_00E4	DSPI DSI Parallel Input Select Register 3 (DSPIx_PISR3)	32	R/W	0000_0000h	on page 1675
0000_00E8	DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPIx_DIMR0)	32	R/W	0000_0000h	on page 1676
0000_00EC	DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPIx_DPIR0)	32	R/W	0000_0000h	on page 1677
0000_00F0	DSPI DSI Serialization Data Register 1 (DSPIx_SDR1)	32	R	0000_0000h	on page 1678
0000_00F4	DSPI DSI Alternate Serialization Data Register 1 (DSPIx_ASDR1)	32	R/W	0000_0000h	on page 1679
0000_00F8	DSPI DSI Transmit Comparison Register 1 (DSPIx_COMPR1)	32	R	0000_0000h	on page 1680
0000_00FC	DSPI DSI Deserialization Data Register 1 (DSPIx_DDR1)	32	R	0000_0000h	on page 1681
0000_0100	DSPI DSI Serialization Source Select Register 1 (DSPIx_SSR1)	32	R/W	0000_0000h	on page 1682
0000_0104	DSPI DSI Parallel Input Select Register 4 (DSPIx_PISR4)	32	R/W	0000_0000h	on page 1683
0000_0108	DSPI DSI Parallel Input Select Register 5 (DSPIx_PISR5)	32	R/W	0000_0000h	on page 1685

Table 778. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_010C	DSPI DSI Parallel Input Select Register 6 (DSPIx_PISR6)	32	R/W	0000_0000h	on page 1686
0000_0110	DSPI DSI Parallel Input Select Register 7 (DSPIx_PISR7)	32	R/W	0000_0000h	on page 1687
0000_0114	DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPIx_DIMR1)	32	R/W	0000_0000h	on page 1688
0000_0118	DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPIx_DPIR1)	32	R/W	0000_0000h	on page 1689
0000_011C	DSPI Clock and Transfer Attributes Register Extended 0 (DSPIx_CTARE0)	32	R/W	0000_0001h	on page 1690
0000_0120	DSPI Clock and Transfer Attributes Register Extended 1 (DSPIx_CTARE1)	32	R/W	0000_0001h	on page 1690
0000_0124	DSPI Clock and Transfer Attributes Register Extended 2 (DSPIx_CTARE2)	32	R/W	0000_0001h	on page 1690
0000_0128	DSPI Clock and Transfer Attributes Register Extended 3 (DSPIx_CTARE3)	32	R/W	0000_0001h	on page 1690
0000_012C	DSPI Clock and Transfer Attributes Register Extended 4 (DSPIx_CTARE4)	32	R/W	0000_0001h	on page 1690
0000_0130	DSPI Clock and Transfer Attributes Register Extended 5 (DSPIx_CTARE5)	32	R/W	0000_0001h	on page 1690
0000_0134	DSPI Clock and Transfer Attributes Register Extended 6 (DSPIx_CTARE6)	32	R/W	0000_0001h	on page 1690
0000_0138	DSPI Clock and Transfer Attributes Register Extended 7 (DSPIx_CTARE7)	32	R/W	0000_0001h	on page 1690
0000_013C	DSPI Status Register Extended (DSPIx_SREX)	32	R	0000_0000h	on page 1691

48.2.10.1 DSPI Module Configuration Register (DSPIx_MCR)

The DSPIx_MCR contains bits to configure various attributes associated with DSPI operations. The HALT and MDIS bits can be changed at any time, but they only take effect on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed while the DSPI is in the Running state.

Figure 733. DSPI Module Configuration Register (DSPIx_MCR)

Address: Base + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSTR	CONT_SCKE	DCONF		FRZ	MTFE	PCSSE	ROOE	PCISIS[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DOZE	MDIS	DIS_TXF	DIS_RXF	0	0	SMPL_PT		0	0	0	0	XSPI	FCPCS	PES	HALT
W					CLR_TXF	CLR_RXF										
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 779. DSPIx_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Configures the DSPI for either master mode or slave mode. 0 DSPI is in slave mode. 1 DSPI is in master mode.
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0 Continuous SCK disabled. 1 Continuous SCK enabled.
2–3 DCONF	DSPI Configuration Selects among the different configurations of the DSPI. 00 SPI 01 DSI 10 CSI 11 Reserved
4 FRZ	Freeze Enables the DSPI transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in debug mode. 1 Halt serial transfers in debug mode.

Table 779. DSPIx_MCR field descriptions (continued)

Field	Description
5 MTFE	Modified Timing Format Enable Enables a modified transfer format to be used. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS[5]/ PCSS to operate as a PCS Strobe output signal. 0 PCS[5]/ is used as the Peripheral Chip Select[5] signal. 1 PCS[5]/PCSS is used as an active-low PCS Strobe signal.
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the DSPI to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8–15 PC SIS[7:0]	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
16 DOZE	Doze Enable Provides support for an externally controlled Doze mode power-saving mechanism. 0 Doze mode has no effect on DSPI. 1 Doze mode disables DSPI.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory-mapped logic in the DSPI effectively putting the DSPI in a software controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 1. When DSPI is used in Slave Mode, it is recommended to leave this bit set to '0', since a slave doesn't have control over master transactions. 0 Enable DSPI clocks. 1 Allow external logic to disable DSPI clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the DSPI (both TXFIFO and CMD FIFO) operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Tx FIFO is enabled. 1 Tx FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Rx FIFO is enabled. 1 Rx FIFO is disabled.

Table 779. DSPIx_MCR field descriptions (continued)

Field	Description
20 CLR_TXF	Clear TX FIFO Flushes the TX FIFO and the CMD FIFO. Writing a '1' to CLR_TXF clears the TX FIFO and CMD FIFO Counters. The CLR_TXF bit is always read as zero. 0 Do not clear the Tx FIFO and CMD FIFO counters. 1 Clears the Tx FIFO and CMD FIFO counters.
21 CLR_RXF	Flushes the RX FIFO. Writing a '1' to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the Rx FIFO counter. 1 Clear the Rx FIFO counter.
22–23 SMPL_PT	Sample Point Controls when the DSPI master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTAR is 0. 00 0 protocol clocks between SCK edge and SIN sample 01 1 protocol clock between SCK edge and SIN sample 10 2 protocol clocks between SCK edge and SIN sample 11 Reserved
24–27 Reserved	This read-only bitfield is reserved and always has the value zero.
28 XSPI	Extended SPI Mode This bit enables usage of CTARE (Command and Transfer Attribute Register Extended) Registers. CTARE registers allow the user to send up to 32-bit SPI frames. Command Cycling is also enabled which allows the user to send multiple Data Frames using a single Command Frame. When MCR[DIS_TXF] is asserted, the Extended SPI Mode cannot be used to transmit SPI frames which are more than 16 bits in size. 0 Normal SPI Mode. Frame size can be up to 16 bits. Command Cycling is not available in this mode. 1 Extended SPI Mode. Up to 32-bit SPI Frames along with Command Cycling Enabled
29 FCPCS	Fast Continuous PCS Mode. This bit enables the masking of "After SCK (t_{ASC})" and "PCS to SCK (t_{CSC})" delays when operating in Continuous PCS mode. This masking is not available if Continuous SCK mode is enabled. The individual delay masks are selected via bits MASC and MCSC of the PUSHHR. The firmware should select appropriate masks when providing continuous frames via the PUSHHR. 0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in PUSHHR.
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt Starts and stops DSPI transfers. 0 Start transfers. 1 Stop transfers.

48.2.10.2 DSPI Transfer Count Register (DSPIx_TCR)

The DSPIx_TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the DSPI is in the Running state.

Figure 734. DSPI Transfer Count Register (DSPIx_TCR)

Address: Base + 0x0008

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SPI_TCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 780. DSPIx_TCR field descriptions

Field	Description
0–15 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the DSPI makes. The SPI_TCNT field increments every time the last bit of a SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
16–31 Reserved	This read-only bitfield is reserved and always has the value zero.

48.2.10.3 DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPIx_CTARn)

CTARs are used to define different transfer attributes. Do not write to the CTARs while the DSPI is in the Running state.

In master mode, the CTARs define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 and CTAR1 are used to set the slave transfer attributes.

When the DSPI is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTARs is used. When the DSPI is configured as an SPI bus slave, the CTAR0 register is used.

When the DSPI is configured as a DSI master, the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0), selects which of the CTARs is used. When the DSPI is configured as a DSI bus slave, the CTAR1 register is used.

In CSI Configuration, the transfer attributes are selected based on whether the current frame is SPI data or DSI data. SPI transfers in CSI Configuration follow the protocol described for SPI Configuration, and DSI transfers in CSI Configuration follow the protocol described for DSI Configuration. CSI Configuration is valid only in conjunction with master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks with help of PDT and DT fields

Figure 735. DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPIx_CTARn)

Address: Base + 0x000C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSSCK				ASC				DT				BR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 781. DSPIx_CTARn field descriptions

Field	Description
0 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in Table 782. See the BR field description for details on how to compute the baud rate. Refer to Table 782 for DSPI SCK duty cycle values. 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
1–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. The minimum valid FMSZ field value is 3. There is a constraint on the minimum allowable Frame size, which depends on the Register Read/Write clock to the Protocol Clock Ratio. The minimum Frame size (FMSZ + 1) is decided by the following equation. Upper Ceiling must be applied for non-integer values. $\text{Min. Frame Size} = (4 \times f_p) + (3 \times f_r) / (n \times f_r)$ fp = Protocol Clock Frequency fr = Register Read/Write Clock Frequency n = Multiple of protocol clock required to get Baud Clock - Given by $(\text{PBR} \times \text{BR}) / (1 + \text{DBR})$ However, the minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Note that 'fp' can be equal to, less than or greater than 'fr' purely based on user requirement. When the DSPI operates in TSB mode, the FMSZ field value plus 1 is equal to the data frame bit number, where control of the PCS assertion switches from the DSICR0 to the DSICR1 register. If TSB Mode is operated when DSI is used in 64 bit mode, then DSICR0[FMSZ4] also comes into picture while switching PCS assertion control.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase or TSB mode Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. When the DSPI operates in TSB configuration, this bit should be set to be compliant to the MSC specification. 0 Data is transferred MSB first. 1 Data is transferred LSB first.

Table 781. DSPIx_CTARn field descriptions (continued)

Field	Description
8–9 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer to Section 48.4.5.2, PCS to SCK delay (tCSC) for more details. In the TSB mode, the PCSSCK field has no effect. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
10–11 PASC	After SCK Delay Prescaler Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. In the TSB mode, the PASC field has no effect. Refer to Section 48.4.5.3, After SCK delay (tASC) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12–13 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. In the TSB mode the PDT field defines two MSB bits of the Delay after Transfer. See the DT field description for details on how to compute the Delay after Transfer. Refer to Section 48.4.5.4, Delay after transfer (tDT) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
14–15 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
16–19 CSSCK	PCS to SCK Delay Scaler Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. In the TSB mode the field has no effect. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_p) \times PCSSCK \times CSSCK$ Table 783 lists the delay scaler values. Refer to Section 48.4.5.2, PCS to SCK delay (tCSC) for more details.

Table 781. DSPIx_CTARn field descriptions (continued)

Field	Description
20–23 ASC	After SCK Delay Scaler Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{ASC} = (1/f_P) \times PASC \times ASC$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer to Section 48.4.5.3, After SCK delay (tASC) for more details.
24–27 DT	Delay After Transfer Scaler Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period. The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. In the TSB mode, the Delay after Transfer is equal to a number formed by concatenation of the PDT and DT fields plus 1 of the SCK clock periods. Refer to Section 48.4.5.2, PCS to SCK delay (tCSC) for more details.
28–31 BR	Baud Rate Scaler Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation: $SCK \text{ baud rate} = (f_P/PBR) \times [(1+DBR)/BR]$ Table 784 lists the baud rate scaler values.

Table 782. DSPI SCK duty cycle

DBR	CPHA	PBR	SCK duty cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table 783. Delay scaler encoding

Field value	Delay scaler value
0000	2
0001	4
0010	8
0011	16
0100	32
0101	64
0110	128
0111	256
1000	512
1001	1024
1010	2048
1011	4096
1100	8192
1101	16384
1110	32768
1111	65536

Table 784. DSPI baud rate scaler

CTARn[BR]	Baud rate scaler value
0000	2
0001	4
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

48.2.10.4 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPIx_CTARn_SLAVE)

When the DSPI is configured as an SPI bus slave, the CTAR0 register is used.

When the DSPI is configured as a DSI master, the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0), selects which of the CTARs is used. When the DSPI is configured as a DSI bus slave, the CTAR1 register is used.

In CSI Configuration, the transfer attributes are selected based on whether the current frame is SPI data or DSI data. SPI transfers in CSI Configuration follow the protocol described for SPI Configuration, and DSI transfers in CSI Configuration follow the protocol described for DSI Configuration. CSI Configuration is valid only in conjunction with master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks with help of PDT and DT fields.

Figure 736. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPIx_CTARn_SLAVE)

Address: Base + 0x000C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FMSZ				CPOL	CPHA	PE	PP	FMSZ[5]	0	0	0	0	0	0	0
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 785. DSPIx_CTARn_SLAVE field descriptions

Field	Description
0–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the concatenated field {CTARn_SLAVE[FMSZ[5]], CTARn_SLAVE[FMSZ]} value plus 1. The minimum valid FMSZ field value is 3 (when FMSZ6 is 0).
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.

Table 785. DSPIx_CTARn_SLAVE field descriptions (continued)

Field	Description
6 CPHA	Clock Phase or TSB mode Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode, the bit value is ignored and the transfers are done as the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9 FMSZ[5]	MSB of Frame Size when DSI is used in 64-bit Mode The number of bits transferred per frame is equal to the concatenated field {CTARn_SLAVE[FMSZ[5]], CTARn_SLAVE[FMSZ]} value plus 1.
10–31 Reserved	This read-only bitfield is reserved and always has the value zero.

48.2.10.5 DSPI Status Register (DSPIx_SR)

SR contains status and flag bits. The bits reflect the status of the DSPI and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear most of the flag bits in the SR by writing a '1' to them. Writing a 0 to a flag bit has no effect. This register may not be writable in module disable mode due to the use of power saving mechanisms.

Figure 737. DSPI Status Register (DSPIx_SR)

Address: Base + 0x002C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	SPITCF	EOQF	TFUF	DSITCF	TFFF	BSYF	CMDTCF	DPEF	SPEF	DDIF	RFOF	TFWIF	RFDF	CMDFFF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNTPTTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 786. DSPIx_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a '1' to it. 0 Transfer not complete. 1 Transfer complete.
1 TXRXS	TX and RX Status Reflects the run status of the DSPI. 0 Transmit and receive operations are disabled (DSPI is in stopped state). 1 Transmit and receive operations are enabled (DSPI is in running state).
2 SPITCF	SPI Frame Transfer Complete Flag. This bit functions similar to TCF except that this bit is asserted only on SPI frame transmission completion in CSI Mode. This includes the ITSB Mode and 'TSB in CSI' Mode as well. This flag will not be asserted when DSPI is used in SPI or DSI Modes. 0 SPI frame transfer is not complete. 1 SPI frame transfer is complete.
3 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the DSPI is in master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a '1' to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.

Table 786. DSPIx_SR field descriptions (continued)

Field	Description
4 TFUF	Transmit FIFO Underflow Flag Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for DSPI blocks operating in slave mode and SPI configuration. TFUF is set when the TX FIFO of a DSPI operating in SPI slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing '1' to it. 0 No Tx FIFO underflow. 1 Tx FIFO underflow has occurred.
5 DSITCF	DSI Frame Transfer Complete Flag. This bit functions similar to TCF except that this bit is asserted only on DSI frame transmission completion in CSI Mode. This includes the ITSB Mode and 'TSB in CSI' Mode as well. This flag will not be asserted when DSPI is used in SPI or DSI Modes. 0 DSI frame transfer is not complete. 1 DSI frame transfer is complete.
6 TFFF	Transmit FIFO Fill Flag Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing '1' to it or by acknowledgement from the DMA controller to the TX FIFO full request. 0 Tx FIFO is full. 1 Tx FIFO is not full.
7 BSYF	Busy Flag This bit is valid only when DSPI_MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmitting multiple data frames. This bit is not set for the last Data Frame of a Cyclic command Transfer or when DSPI_CTARE[DTCP] = 1. Refer to Section 48.4.2.5, Command First In First Out (CMD FIFO) buffering mechanism for more details. 0 No Cyclic Command Transfer in Progress. 1 Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for current cyclic command transfer.
8 CMDTCF	Command Transfer Complete Flag Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when DSPI_CTARE[DTCP] = 1. CMDTCF remains set until it is cleared by writing a '1' to it. 0 Data Transfer by current Command not complete. 1 Data Transfer by current Command complete.
9 DPEF	DSI Parity Error Flag Indicates that a DSI frame with parity error had been received. The bit remains set until it is cleared by writing a '1' to it. 0 No parity error. 1 Parity error has occurred.
10 SPEF	SPI Parity Error Flag Indicates that a SPI frame with parity error had been received. The bit remains set until it is cleared by writing a '1' to it. 0 No parity error. 1 Parity error has occurred.

Table 786. DSPIx_SR field descriptions (continued)

Field	Description
11 DDIF	DSI Data Received with Active Bits Indicates that a DSI frame had been received with bits selected by DIMR with active polarity, defined by the DPIR. The bit remains set until it is cleared by writing a '1' to it. 0 No DSI data with active bits was received. 1 DSI data with active bits was received.
12 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a '1' to it. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 TFIWF	Transmit FIFO Invalid Write Flag Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. 0 No Invalid Data present in TX FIFO 1 Invalid Data present in TX FIFO since CMD FIFO is empty
14 RFDF	Receive FIFO Drain Flag Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 Rx FIFO is empty. 1 Rx FIFO is not empty.
15 CMDFFF	Command FIFO Fill Flag Provides a method for the DSPI to request more entries to be added to the CMD FIFO. The CMDFFF bit is set while the CMD FIFO is not full. The CMDFFF is cleared by writing a '1' to it or by acknowledgement from the DMA controller to the CMD FIFO full request. 0 CMD FIFO is full. 1 CMD FIFO is not full.
16–19 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the data part of PUSHR is written. The TXCTR is decremented every time the SPI data is transferred to the shift register.
20–23 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO Entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
24–27 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28–31 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

48.2.10.6 DSPI DMA/Interrupt Request Select and Enable Register (DSPIx_RSER)

The DSPIx_RSER controls DMA and interrupt requests. Do not write to the RSER while the DSPI is in the Running state.

Figure 738. DSPI DMA/Interrupt Request Select and Enable Register (DSPIx_RSER)

Address: Base + 0x0030

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF_RE	CMDFFF_RE	SPITCF_RE	EOQF_RE	TFUF_RE	DSITCF_RE	TFFF_RE	TFFF_DIRS	CMDTCF_RE	DPEF_RE	SPEF_RE	DDIF_RE	RFOF_RE	TFWF_RE	RFDF_RE	RFDF_DIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDFFF_DIRS	DDIF_DIRS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 787. DSPIx_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable Enables the CMDFFF flag in the SR to generate a request. The CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 CMDFFF interrupts or DMA requests are disabled. 1 CMDFFF interrupts or DMA requests are enabled.
2 SPITCF_RE	SPI Frame Transmission Complete Request Enable. The SPITCF_RE bit enables SPITCF flag in the SR to generate an interrupt request. 0 SPITCF interrupt requests are disabled. 1 SPITCF interrupt requests are enabled.
3 EOQF_RE	DSPI Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request. 0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.

Table 787. DSPIx_RSER field descriptions (continued)

Field	Description
5 DSITCF_RE	DSI Frame Transmission Complete Request Enable. The DSITCF_RE bit enables DSITCF flag in the SR to generate an interrupt request. 0 DSITCF interrupt requests are disabled. 1 DSITCF interrupt requests are enabled.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the TFFF flag bit in the SR is set and the TFFF_RE bit in the RSER is set, this bit selects between generating an interrupt request or a DMA request. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the SR to generate an interrupt request. 0 CMDTCF interrupt requests are disabled. 1 CMDTCF interrupt requests are enabled.
9 DPEF_RE	DSI Parity Error Request Enable Enables the DPEF flag in the SR to generate an interrupt request. 0 DPEF interrupt requests are disabled. 1 DPEF interrupt requests are enabled.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the SR to generate an interrupt request. 0 SPEF interrupt requests are disabled. 1 SPEF interrupt requests are enabled.
11 DDIF_RE	DSI data received with active bits Request Enable Enables the DDIF flag in the SR to generate an interrupt requests. 0 DDIF interrupt requests are disabled. 1 DDIF interrupt requests are enabled.
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable Enables the TFIWF flag in the SR to generate an interrupt request. 0 TFIWF interrupt requests are disabled. 1 TFIWF interrupt requests are enabled.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt or DMA requests are disabled 1 RFDF interrupt or DMA requests are enabled

Table 787. DSPIx_RSER field descriptions (continued)

Field	Description
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
16 CMDFFF_DIRS	Command FIFO Fill DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. When the CMDFFF flag bit in the SR is set, and the CMDFFF_RE bit in the RSER is set, the CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
17 DDIF_DIRS	DSI data received with active bits - DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. When the DDIF flag bit in the SR is set, and the DDIF_RE bit in the RSER is set, the DDIF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
18–31 Reserved	This read-only bitfield is reserved and always has the value zero.

48.2.10.7 DSPI PUSH FIFO Register In Master Mode (DSPIx_PUSHR)

PUSHR provides the means to write to the TX FIFO and CMD FIFO. Data written to this register is transferred to the TX FIFO and/or CMD FIFO. Eight- or sixteen-bit write accesses to the Data Field of PUSHR transfers the 16-bit Data field of PUSHR to the TX FIFO. Write accesses to the Command Field of PUSHR transfers the 16-bit Command Field of PUSHR to the CMD FIFO. The register structure is different in master and slave modes. In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO. In slave mode, CMD FIFO is unused and hence the 16-bit Command Field of PUSHR is reserved.

Unless Extended SPI Mode is being used, the TX FIFO and CMD FIFO must be filled simultaneously. In other words, write accesses must be given to both Data and Command fields of PUSHR for every PUSHR operation performed when MCR[XSPI] is 0. When Extended SPI Mode is disabled, since both TX FIFO and CMD FIFO are written to and read from simultaneously, they can be thought of as a single 32-bit FIFO.

When Extended SPI Mode is enabled, TX FIFO and CMD FIFO can be written independently. A PUSHR Read Operation returns the topmost TX FIFO and CMD FIFO entries concatenated.

When DSPI Module is disabled, any writes to this register will not update the FIFO. Hence any reads performed during Module disable mode will return the last PUSHR write performed when Module was enabled.

Figure 739. DSPI PUSH FIFO Register In Master Mode (DSPIx_PUSHR)

Address: Base + 0x0034																Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	CONT	CTAS			EOQ	CTCNT	PE_MASC	PP_MCSC	PCS[7:0]																						
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	TXDATA																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

Table 788. DSPIx_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable Selects a Continuous Selection Format. The bit is used in SPI master mode. The bit enables the selected PCS signals to remain asserted between transfers. PUSHR[EOQ] and PUSHR[CONT] bits cannot be asserted together in any Command Frame, since EOQ signifies the last frame in a queue and the Last frame in a Continuous Transfer must have PUSHR[CONT] as 0. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1–3 CTAS	Clock and Transfer Attributes Select. Selects which CTAR (and corresponding CTARE register) to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI slave mode, CTAR0 is used. See the Chip Configuration chapter to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0/CTARE0 001 CTAR1/CTARE1 010 CTAR2/CTARE2 011 CTAR3/CTARE3 100 CTAR4/CTARE4 101 CTAR5/CTARE5 110 CTAR6/CTARE6 111 CTAR7/CTARE7
4 EOQ	End Of Queue Host software uses this bit to signal to the DSPI that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter. Clears the SPI_TCNT field in the TCR. The SPI_TCNT field is cleared before the DSPI starts transmitting the current SPI frame. Hence the current SPI frame will cause this counter to increment by 1. 0 Do not clear the TCR[SPI_TCNT] field. 1 Clear the TCR[SPI_TCNT] field.
6 PE_MASC	Parity Enable or Mask t_{ASC} delay in the current frame PE – This bit enables parity bit transmission and parity reception check for the SPI frame. MASC - The current frame has the “after SCK” delay masked if this bit is asserted. See Section 48.4.6.6, Fast Continuous Selection Format for more details. <i>Note: This bit is used as Mask t_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set.</i> 0 PE - No parity bit included/checked. MASC - TASC delay is not masked and the current frame has the after SCK delay. 1 PE - Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC - TASC delay is masked in the current frame.

Table 788. DSPIx_PUSHR field descriptions (continued)

Field	Description
7 PP_MCSC	Parity Polarity or Mask t_{CSC} delay in the next frame PP - It controls the polarity of the parity bit transmitted and checked. MCSC - The next frame has the "PCS to SCK" delay masked if this bit is asserted. See Section 48.4.6.6, Fast Continuous Selection Format for more details. <i>Note: This bit is used as Mask t_{CSC} in the Fast Continuous PCS mode when MCR[FCPCS] is set.</i> 0 PP - Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. MCSC - TCSC delay is not masked and the next frame has the PCS to SCK delay. 1 PP - Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC - TCSC delay is masked in the next frame.
8–15 PCS[7:0]	Select which PCS signals are to be asserted for the transfer. Refer to the chip configuration chapter for the number of PCS signals used in this MCU. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

48.2.10.8 DSPI PUSH FIFO Register In Slave Mode (DSPIx_PUSHR_SLAVE)

PUSHR provides the means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. Eight or 16-bit write accesses to the Data Field of PUSHR transfers the 16-bit Data Field of PUSHR to the TX FIFO. The register structure is different in master and slave modes. In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO. In slave mode, CMD FIFO is unused and hence the 16-bit Command Field of PUSHR is reserved.

In Slave mode, up to 32-bit SPI frames may be queued for transmission/reception by enabling MCR[XSPI] mode.

Figure 740. DSPI PUSH FIFO Register In Slave Mode (DSPIx_PUSHR_SLAVE)

Address: Base + 0x0034

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 789. DSPIx_PUSHR_SLAVE field descriptions

Field	Description
0–15 Reserved	This read-only bitfield is reserved and always has the value zero.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred.

48.2.10.9 DSPI POP FIFO Register (DSPIx_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Figure 741. DSPI POP FIFO Register (DSPIx_POPR)

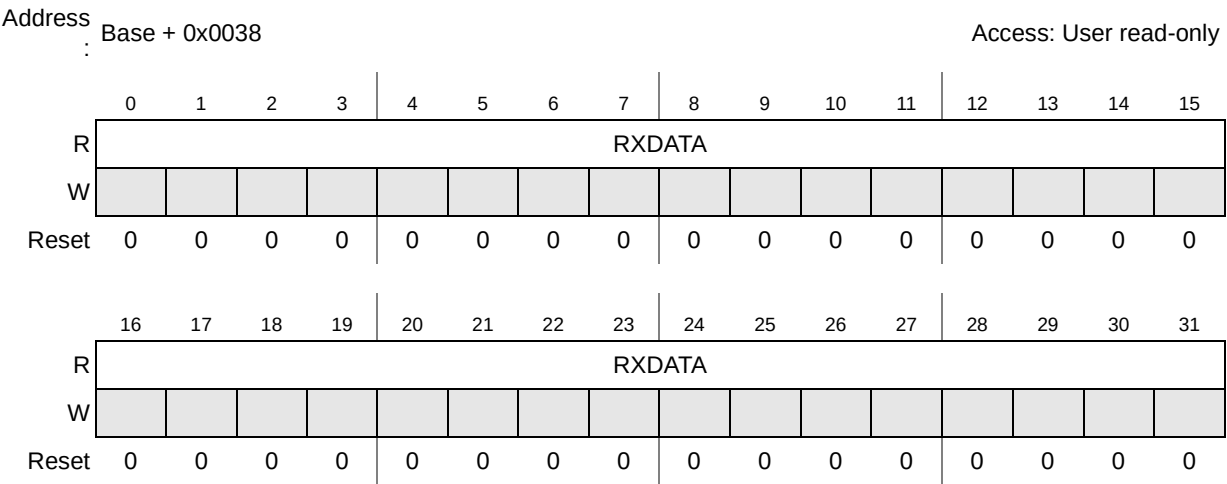


Table 790. DSPIx_POPR field descriptions

Field	Description
0–31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

48.2.10.10 DSPI Transmit FIFO Registers (DSPIx_TXFRn)

TXFRn provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRn registers does not alter the state of the TX FIFO. Reading DSPIx_TXFRn registers is invalid for a DSPI Master when being used in the Extended SPI (XSPI) Mode.

Figure 742. DSPI Transmit FIFO Registers (DSPIx_TXFRn)

Address: Base + 0x003C

Access: User read-only

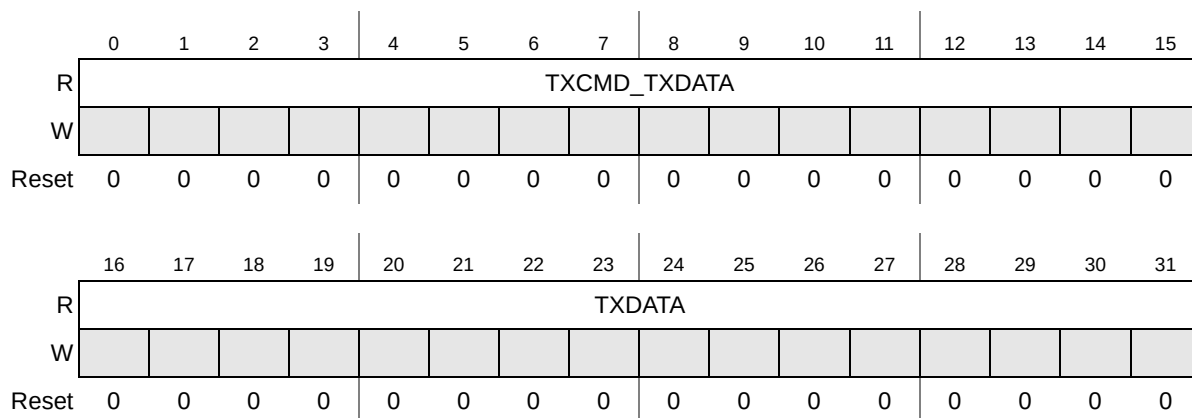


Table 791. DSPIx_TXFRn field descriptions

Field	Description
0–15 TXCMD_TXDATA	Transmit Command or Transmit Data In master mode, the TXCMD field contains the command that sets the transfer attributes for the SPI data. In slave mode, the TXDATA contains 16 MSB bits of the SPI data to be shifted out.
16–31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

48.2.10.11 DSPI Receive FIFO Registers (DSPIx_RXFRn)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFRs are read-only. Reading the RXFRn registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

Figure 743. DSPI Receive FIFO Registers (DSPIx_RXFRn)

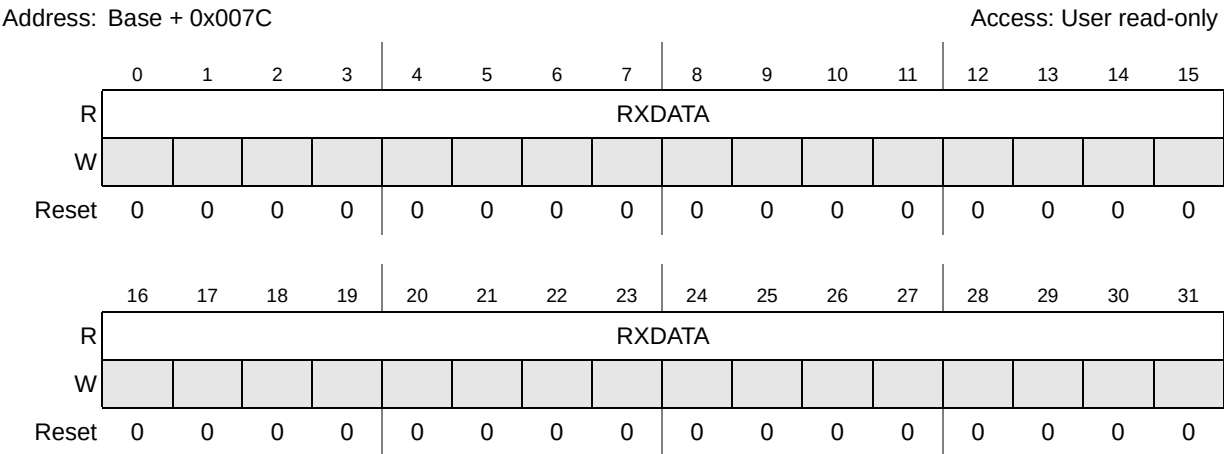


Table 792. DSPIx_RXFRn field descriptions

Field	Description
0–31 RXDATA	Receive Data Contains the received SPI data.

48.2.10.12 DSPI DSI Configuration Register 0 (DSPIx_DSICR0)

DSICR0 selects various attributes associated with DSI and CSI Configurations. Do not write to the DSICR0 while the DSPI is in the Running state.

Figure 744. DSPI DSI Configuration Register 0 (DSPIx_DSICR0)

Address: Base + 0x00BC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MTOE	FMSZ[4]	MTOCNT[5:0]						FMSZ[5]	0	ITSB	TSBC	TXSS	TPOL	TRRE	CID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCONT	DSICTAS[2:0]				DMS	PES	PE	PP	DPCSx						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 793. DSPIx_DSICR0 field descriptions

Field	Description
0 MTOE	Multiple Transfer Operation Enable Enables multiple DSPIs to be connected in a parallel or serial configuration. The MTOE and TSB bits should not be set simultaneously. 0 Multiple transfer operation disabled. 1 Multiple transfer operation enabled.
1 FMSZ[4]	MSB of the frame size in master mode when DSI is used in 32-bit mode. If the bit is set, 16 is added to the frame size, as defined by the CTARn[FMSZ] field. The CTARx register is selected by the DSICTAS field.
2–7 MTOCNT[5:0]	Multiple Transfer Operation Count Selects the number of bits to be shifted out during a transfer in Multiple Transfer Operation. The field sets the number of SCK cycles that the bus master must generate to complete the transfer. The number of SCK cycles used is one more than the value in the MTOCNT field. The number of SCK cycles defined by MTOCNT must be equal to or greater than the frame size. When TSBC is set, MTOCNT has no effect.
8 FMSZ[5]	MSB of the frame size in master mode when DSI is used in 64-bit mode. If the bit is set, 32 is added to the frame size, as defined by the concatenation of {DSICR0[FMSZ[4]], CTARn[FMSZ]} fields. The CTARx register is selected by the DSICTAS field. Hence the final concatenated frame size using this bit would be {DSICR0[FMSZ[5]], DSICR0[FMSZ[4]], CTARn[FMSZ]}. This field is only valid when DSICR1[DSI64] is enabled.
9 Reserved	This read-only bit is reserved. Note that this bit must always be written as zero.

Table 793. DSPIx_DSICR0 field descriptions (continued)

Field	Description
10 ITSB	Interleaved TSB mode. This bit enables the Interleaved TSB mode of operation. When enabled, there is no priority among frames from the SPI/DSI. DSI frames are always sent when either the previous transmission was a SPI frame or if the TX_FIFO is empty. Frames are transmitted on every trigger whose source is selected by the TRG bit setting. The ITSB mode requires the DSICR0[TSBC] bit to be set and should be written only when MCR[HALT] bit is asserted. If ITSB bit is set without setting DSICR0[TSBC], then the DSPI operates in the Normal Mode depending on MCR[DCONF] bit. See Section 48.4.10, Interleaved TSB (ITSB) Mode for more details. 0 Interleaved mode is disabled. 1 Interleaved mode is enabled.
11 TSBC	Timed Serial Bus Configuration. Enables the Timed Serial Bus Configuration, which allows up to 64-bit data to be used. It also allows t_{DT} to be programmable. 0 Timed Serial Bus Configuration disabled. 1 Timed Serial Bus Configuration enabled.
12 TXSS	Transmit Data Source Select. Selects the source of data to be serialized. The source can be either data from host software written to the DSPI DSI Alternate Serialization Data Register (ASDR1/0) or parallel input pin states latched into the DSPI DSI Serialization Data Register (SDR1/0). 0 Source of serialized data is the SDR. 1 Source of serialized data is the ASDR.
13 TPOL	Trigger Polarity Selects the active edge of the hardware trigger input signal (HT), initiating DSI frames transfer. 0 Falling edge initiates a transfer. 1 Rising edge initiates a transfer.
14 TRRE	Trigger Reception Enable Enables the DSPI to initiate a DSI frames transfer with an external trigger signal. 0 Trigger signal reception disabled. 1 Trigger signal reception enabled.
15 CID	Change In Data Transfer Enable Enables a change in serialization data to initiate a DSI frames transfer in DSI and CSI configurations. When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out.
16 DCONT	DSI Continuous Peripheral Chip Select Enable Enables the PCS signals to remain asserted between transfers. The DCONT bit affects the PCS signals only in DSI master mode. When the TSBC bit is set, DCONT has no effect. 0 Return Peripheral Chip Select signals to their inactive state after the transfer is complete. 1 Keep Peripheral Chip Select signals asserted after transfer is complete.
17–19 DSICTAS[2:0]	DSI Clock and Transfer Attributes Select Selects the CTAR to use to provide transfer attributes for DSI frames. The DSICTAS field is used in DSI master mode. In DSI slave mode, CTAR1 is always selected.
20 DMS	Data Match Stop If set, stops DSI frame transmissions if DDIF flag is set in the SR. 0 DDIF flag does not have an effect on DSI frame transmissions. 1 DDIF flag stops DSI frame transmissions.

Table 793. DSPIx_DSICR0 field descriptions (continued)

Field	Description
21 PES	Parity Error Stop If set, stops DSI operation if the parity error happened in the received DSI frame. 0 Parity error does not stop DSI frame transmissions. 1 Parity error stops all DSI frame transmissions.
22 PE	Parity Enable Enables parity bit transmission and parity reception check for the DSI frames. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
23 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[DPEF] bit is set if in the received frame number of 1 bits is odd. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[DPEF] bit is set if in the received frame number of 1 bits is even.
24–31 DPCSx	DSI Peripheral Chip Select 0–7 Selects which of the PCS signals to assert during a DSI master mode transfer. 0 Negate PCS[x]. 1 Assert PCS[x].

48.2.10.13 DSPI DSI Serialization Data Register 0 (DSPIx_SDR0)

SDR0 contains the states of the 32 LSB parallel input signals. The states of these parallel input signals are latched into the SDR0 on the rising edge of every protocol clock. The SDR0 is read-only. When the TXSS bit in the DSICR0 is cleared, the data in the SDR0 is used as the source of the DSI frames. When DSICR1[DSI64E] is enabled, SDR1 forms the MSB half and SDR0 forms the LSB half of the 64-bit Serialization Data Register.

Figure 745. DSPI DSI Serialization Data Register 0 (DSPIx_SDR0)

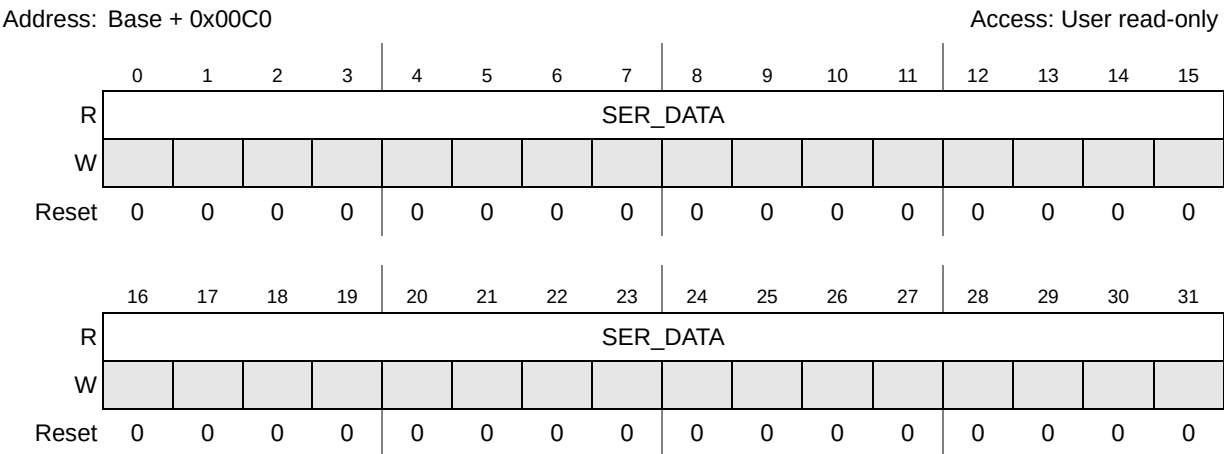


Table 794. DSPIx_SDR0 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the signal states of the 32 LSB parallel input signals.



48.2.10.14 DSPI DSI Alternate Serialization Data Register 0 (DSPIx_ASDR0)

ASDR0 is used by host software to write the 32 LSB of the data to be serialized. When the TXSS bit in the DSICR0 is set, the data in the ASDR0 is the source of the DSI frames. Writes to the ASDR0 take effect on the next frame boundary. When DSICR1[DSI64E] is enabled, ASDR1 forms the MSB half while ASDR0 forms the LSB half of the 64-bit Alternate Serialization Data Register.

Figure 746. DSPI DSI Alternate Serialization Data Register 0 (DSPIx_ASDR0)

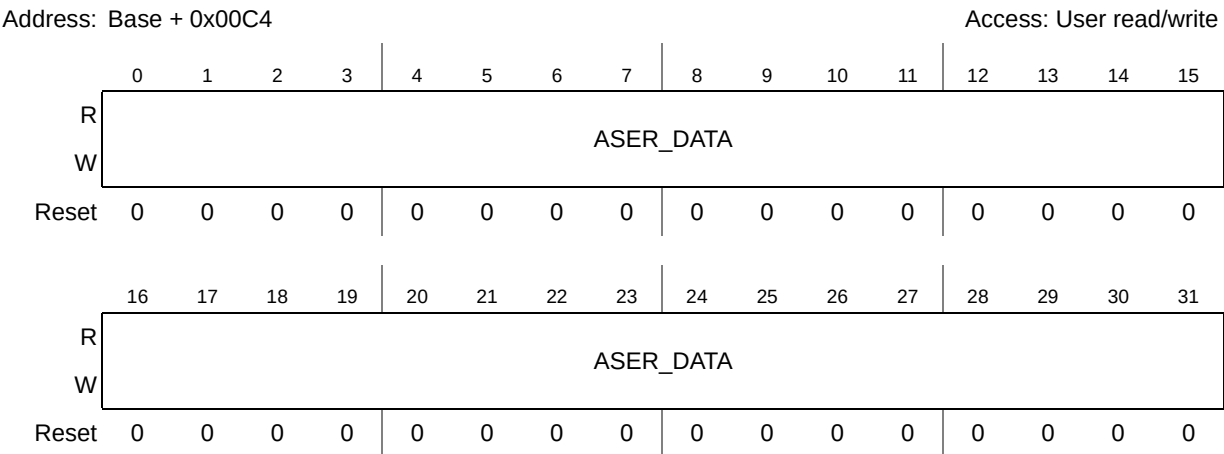


Table 795. DSPIx_ASDR0 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the alternate 32 LSB of data to be serialized.

48.2.10.15 DSPI DSI Transmit Comparison Register 0 (DSPIx_COMPR0)

COMPR0 holds a copy of the 32 LSB of last transmitted DSI data. COMPR0 is read-only. DSI data is transferred to this register as it is loaded into the TX Shift Register. When DSICR1[DSI64E] is enabled, COMPR1 forms the MSB half while COMPR0 forms the LSB half of the 64-bit Transmit Comparison Register

Figure 747. DSPI DSI Transmit Comparison Register 0 (DSPIx_COMPR0)

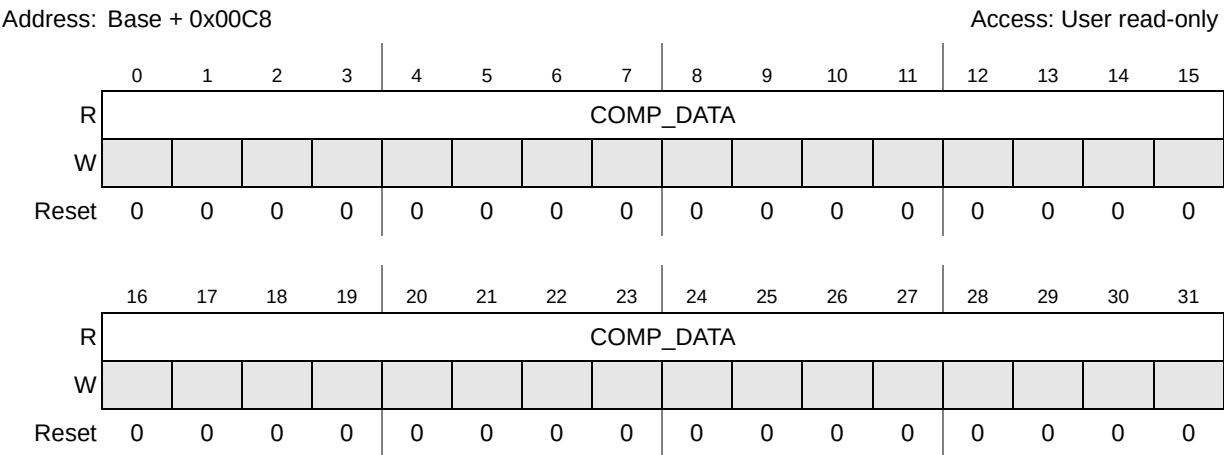


Table 796. DSPIx_COMPR0 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data Holds the 32 LSB of last serialized DSI data.



48.2.10.16 DSPI DSI Deserialization Data Register 0 (DSPIx_DDR0)

DDR0 holds the 32 LSB signal states for the Parallel Output signals. The DDR0 is read-only and host software can read data from incoming DSI frames. When DSICR1[DSI64E] is enabled, DDR1 forms the MSB half while DDR0 forms the LSB half of the 64-bit Deserialization Data Register.

Figure 748. DSPI DSI Deserialization Data Register 0 (DSPIx_DDR0)

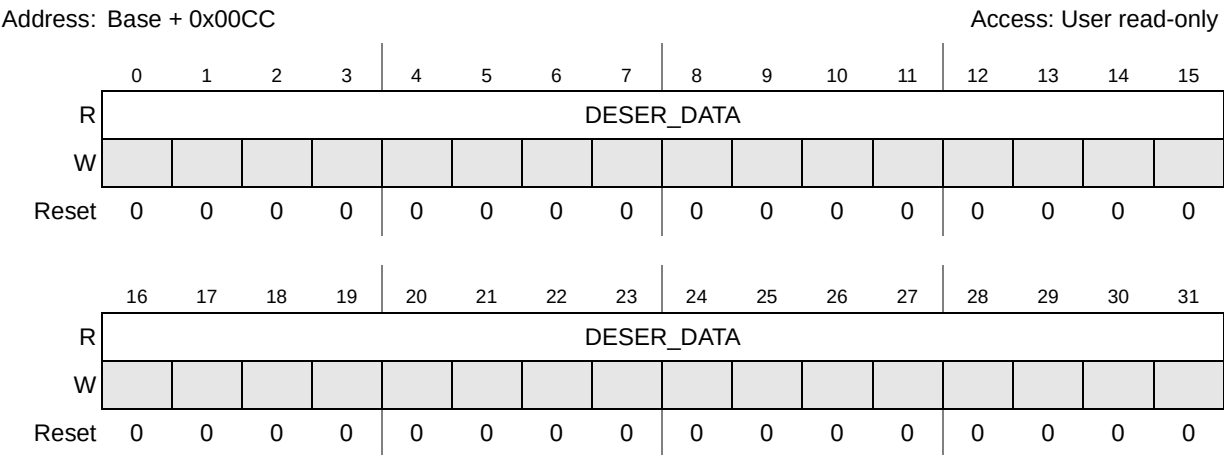


Table 797. DSPIx_DDR0 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data Holds 32 LSB of deserialized data that is presented as signal states to the Parallel Output signals.

48.2.10.17 DSPI DSI Configuration Register 1 (DSPIx_DSICR1)

DSICR1 selects various attributes associated with TSB (and ITSB) Configuration. The user must not write to the DSICR1 while the DSPI is in the Running state.

Figure 749. DSPI DSI Configuration Register 1 (DSPIx_DSICR1)

Address: Base + 0x00D0

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	TSBCNT[5:0]					0	0	0	0	0	0	DSI64E	DSE 1	DSE 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRGPRD								DPCS1_x							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 798. DSPIx_DSICR1 field descriptions

Field	Description
0–1 Reserved	This read-only bitfield is reserved and always has the value zero.
2–7 TSBCNT[5:0]	Timed Serial Bus Operation Count When TSBC is set, TSBCNT defines the length of the data frame. The TSBCNT field valid value is from 3 to 63. The TSBCNT field selects the number of data bits to be shifted out during a transfer in TSB mode. The number of data bits in the data frame is one more than the value in the TSBCNT field.
8–12 Reserved	This read-only bitfield is reserved and always has the value zero.
13 DSI64E	DSI 64-bit Mode Enable When DSI64E is set, DSI Mode operates in 64-bit configuration allowing serialization and deserialization of Data frames up to 64 bits in size. The Extended Registers DSPIx_SDR1, DSPIx_AS DR1, DSPIx_COMPR1, DSPIx_DDR1, DSPIx_SSR1, DSPIx_PISR4-7, DSPIx_DIMR1, DSPIx_DPIR1 are only valid when this Mode is enabled. 0 DSI 64-bit Mode Disabled. DSI Mode operates in the default 32-bit configuration 1 DSI 64-bit Mode Enabled. DSI Mode operates in 64-bit configuration

Table 798. DSPIx_DSICR1 field descriptions (continued)

Field	Description
14 DSE1	Data Select Enable 1 When DSICR0[TBSC] bit is set, the DSE1 bit controls insertion of the zero bit (Data Select) in the middle of the data frame. The insertion bit position is defined by the FMSZ field of CTARn register, selected by the DSICR0[DSICTAS] field. If DSICR1[DSI64] is enabled, then the insertion bit position is defined by concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields. The TSBCNT field value must be greater than the FMSZ field value plus one for proper operation of the DSE1 bit. 0 No Zero bit is inserted in the middle of the data frame. 1 Zero bit is inserted at the middle of the data frame. Total number of bits in the data frame is increased by 1.
15 DSE0	Data Select Enable 0 When DSICR0[TSBC] bit is set, the DSE0 bit controls insertion of the zero bit (Data Select) in the beginning of the Data frame. 0 No Zero bit is inserted in the beginning of the frame. 1 Zero bit is inserted at the beginning of the data frame. Total number of bits in the data frame is increased by 1.
16–23 TRGPRD	Internal Trigger Period for the ITSB mode. When DSICR0[TRG] bit is cleared and ITSB Mode is enabled, this field determines the trigger period for the internal trigger in number of baud rate clock cycles. The trigger period should be calculated as: Trigger Period = 64 bits (DSI frame maximum size) + 2 selection bits (maximum) + t_{PF} (passive frame time) When DSICR1[DSI64E] is disabled, DSI frame maximum size is 32. Refer CTAR[FMSZ] field for minimum frame size allowed. See Interleaved TSB (ITSB) Mode for more details. 0 – 4 These values are invalid since minimum Command frame size is 5. (i.e. 1 selection bit + minimum frame size which is 4) 5 – 255 Number of baud rate clock cycles to be used as the trigger period. The trigger period implemented is equal to TRGPRD + 1.
24–31 DPCS1_x	DSI Peripheral Chip Select 0–7 These bits define the PCSs to assert for the second part of the DSI frame when operating in TSB (and ITSB) configuration with dual receiver. The DPCS1 bits select which of the PCS signals to assert during the second part of the DSI frame. The DPCS1 bits control the assertions of the PCS signals only in TSB and ITSB mode. 0 Negate PCS[x]. 1 Assert PCS[x].

48.2.10.18 DSPI DSI Serialization Source Select Register 0 (DSPIx_SSR0)

SSR0 is used to create a combined frame for transmission that contains bits from ASDR0 and from SDR0. Each bit in the SSR0 register selects a corresponding bit to be serialized. When DSICR0[TXSS] is set, the SSR0 register value has no effect. When DSICR1[DSI64E] is enabled, SSR1 forms the MSB half while SSR0 forms the LSB half of the 64-bit Serialization Source Select Register.

Figure 750. DSPI DSI Serialization Source Select Register 0 (DSPIx_SSR0)

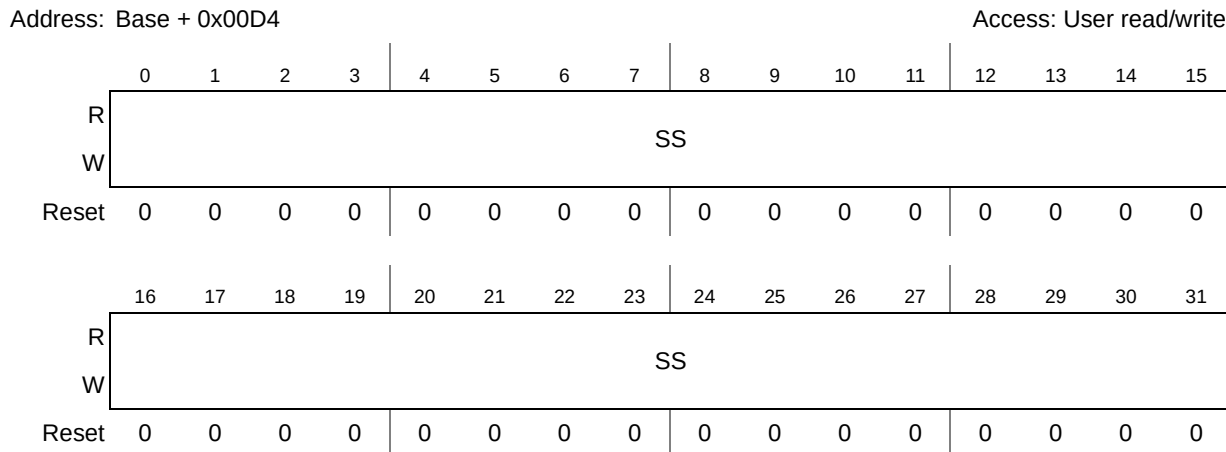


Table 799. DSPIx_SSR0 field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 LSB of DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. 0The bit in the transmitted frame is taken from the parallel input pin. 1The bit in the transmitted frame is taken from the ASDR0 register

48.2.10.19 DSPI DSI Parallel Input Select Register 0 (DSPIx_PISR0)

PISR0–3 are used to select each data bit for the transmitted frame from 16 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31 (bit 0 is taken from parallel input 63 if DSICR1[DSI64E] is enabled). When the IPS0 is equal to +1, then bit 0 in the frame is taken from parallel input 1 and so on.

Note that PISR0–3 only preselect the parallel input pins. The final selection to the transmitted frame is made with the DSPI_SSR1/0 register bits or the DSICR0[TXSS] bit.

When DSICR1[DSI64E] is enabled, PISR4-7 Registers are used to handle the extra 32 bits (32 MSB pins of the 64 parallel input pins).

Figure 751. DSPI DSI Parallel Input Select Register 0 (DSPIx_PISR0)

Address: Base + 0x00D8

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPS7				IPS6				IPS5				IPS4			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS3				IPS2				IPS1				IPS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 800. DSPIx_PISR0 field descriptions

Field	Description
0–3 IPS7	Input Pin Select 7 Selects the parallel input pin for transmitted frame bit 7.
4–7 IPS6	Input Pin Select 6 Selects the parallel input pin for transmitted frame bit 6.
8–11 IPS5	Input Pin Select 5 Selects the parallel input pin for transmitted frame bit 5.
12–15 IPS4	Input Pin Select 5 Selects the parallel input pin for transmitted frame bit 4.
16–19 IPS3	Input Pin Select 3 Selects the parallel input pin for transmitted frame bit 3.

Table 800. DSPIx_PISR0 field descriptions (continued)

Field	Description
20–23 IPS2	Input Pin Select 2 Selects the parallel input pin for transmitted frame bit 2.
24–27 IPS1	Input Pin Select 1 Selects the parallel input pin for transmitted frame bit 1.
28–31 IPS0	Input Pin Select 0 Selects the parallel input pin for transmitted frame bit 0.

48.2.10.20 DSPI DSI Parallel Input Select Register 1 (DSPIx_PISR1)

Figure 752. DSPI DSI Parallel Input Select Register 1 (DSPIx_PISR1)

Address: Base + 0x00DC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPS15				IPS14				IPS13				IPS12			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS11				IPS10				IPS9				IPS8			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 801. DSPIx_PISR1 field descriptions

Field	Description
0–3 IPS15	Input Pin Select 15 Selects the parallel input pin for transmitted frame bit 15.
4–7 IPS14	Input Pin Select 14 Selects the parallel input pin for transmitted frame bit 14.
8–11 IPS13	Input Pin Select 13 Selects the parallel input pin for transmitted frame bit 13.
12–15 IPS12	Input Pin Select 12 Selects the parallel input pin for transmitted frame bit 12.
16–19 IPS11	Input Pin Select 11 Selects the parallel input pin for transmitted frame bit 11.
20–23 IPS10	Input Pin Select 10 Selects the parallel input pin for transmitted frame bit 10.
24–27 IPS9	Input Pin Select 9 Selects the parallel input pin for transmitted frame bit 9.
28–31 IPS8	Input Pin Select 8 Selects the parallel input pin for transmitted frame bit 8.

48.2.10.21 DSPI DSI Parallel Input Select Register 2 (DSPIx_PISR2)

Figure 753. DSPI DSI Parallel Input Select Register 2 (DSPIx_PISR2)

Address: Base + 0x00E0

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPS23				IPS22				IPS21				IPS20			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS19				IPS18				IPS17				IPS16			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 802. DSPIx_PISR2 field descriptions

Field	Description
0–3 IPS23	Input Pin Select 23 Selects the parallel input pin for transmitted frame bit 23.
4–7 IPS22	Input Pin Select 22 Selects the parallel input pin for transmitted frame bit 22.
8–11 IPS21	Input Pin Select 21 Selects the parallel input pin for transmitted frame bit 21.
12–15 IPS20	Input Pin Select 20 Selects the parallel input pin for transmitted frame bit 20.
16–19 IPS19	Input Pin Select 19 Selects the parallel input pin for transmitted frame bit 19.
20–23 IPS18	Input Pin Select 18 Selects the parallel input pin for transmitted frame bit 18.
24–27 IPS17	Input Pin Select 17 Selects the parallel input pin for transmitted frame bit 17.
28–31 IPS16	Input Pin Select 16 Selects the parallel input pin for transmitted frame bit 16.

48.2.10.22 DSPI DSI Parallel Input Select Register 3 (DSPIx_PISR3)

Figure 754. DSPI DSI Parallel Input Select Register 3 (DSPIx_PISR3)

Address: Base + 0x00E4

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPS31				IPS30				IPS29				IPS28			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS27				IPS26				IPS25				IPS24			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 803. DSPIx_PISR3 field descriptions

Field	Description
0–3 IPS31	Input Pin Select 31 Selects the parallel input pin for transmitted frame bit 31.
4–7 IPS30	Input Pin Select 30 Selects the parallel input pin for transmitted frame bit 30.
8–11 IPS29	Input Pin Select 29 Selects the parallel input pin for transmitted frame bit 29.
12–15 IPS28	Input Pin Select 28 Selects the parallel input pin for transmitted frame bit 28.
16–19 IPS27	Input Pin Select 27 Selects the parallel input pin for transmitted frame bit 27.
20–23 IPS26	Input Pin Select 26 Selects the parallel input pin for transmitted frame bit 26.
24–27 IPS25	Input Pin Select 25 Selects the parallel input pin for transmitted frame bit 25.
28–31 IPS24	Input Pin Select 24 Selects the parallel input pin for transmitted frame bit 24.

48.2.10.23 DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPIx_DIMR0)

DIMR0 selects bits in the 32 LSB of received DSI frame to be checked to generate the DDI interrupt. When DSICR1[DSI64E] is enabled, DIMR1 forms the MSB half while DIMR0 forms the LSB half of the 64-bit Deserialized Data Interrupt Mask Register.

Figure 755. DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPIx_DIMR0)

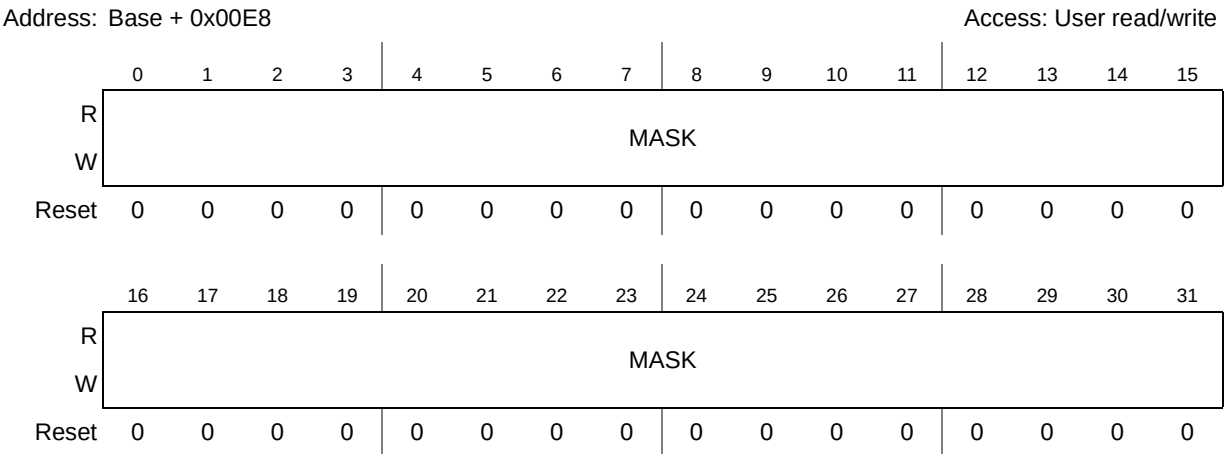


Table 804. DSPIx_DIMR0 field descriptions

Field	Description
0–31 MASK	Mask Defines which bits in the 32 LSB of received deserialization data should be checked to produce the Deserialized Data Interrupt (DDI). 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.

48.2.10.24 DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPIx_DPIR0)

DPIR0 defines which data bit value in the 32 LSB of received DSI frame generates the DDI interrupt. When DSICR1[DSI64E] is enabled, DPIR1 forms the MSB half while DPIR0 forms the LSB half of the 64-bit Deserialized Data Polarity Interrupt Register.

Figure 756. DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPIx_DPIR0)

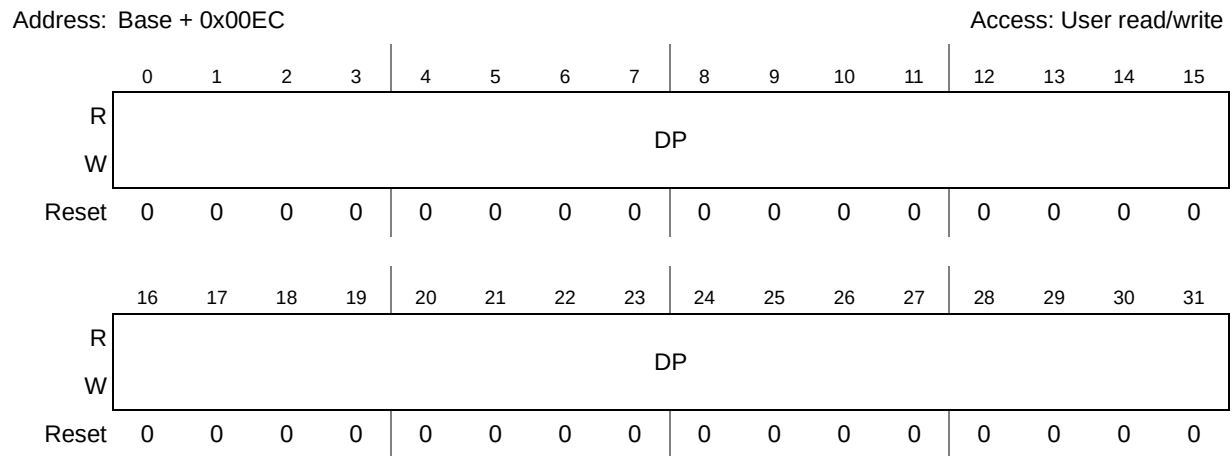


Table 805. DSPIx_DPIR0 field descriptions

Field	Description
0–31 DP	Data Polarity Defines which value of the 32 LSB of received deserialization data sets the SR[DDIF] bit. 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

48.2.10.25 DSPI DSI Serialization Data Register 1 (DSPIx_SDR1)

SDR1 contains the states of the 32 MSB parallel input signals. The states of these parallel input signals are latched into the SDR1 on the rising edge of every protocol clock. The SDR1 is read-only. When the TXSS bit in the DSICR0 is cleared, the data in the SDR1 and SDR0 is used as the source of the DSI frames. The concatenation of {DSPIx_SDR1, DSPIx_SDR0} provides the 64-bit data to be serialized. This register is valid only when DSICR1[DSI64E] is enabled.

Figure 757. DSPI DSI Serialization Data Register 1 (DSPIx_SDR1)

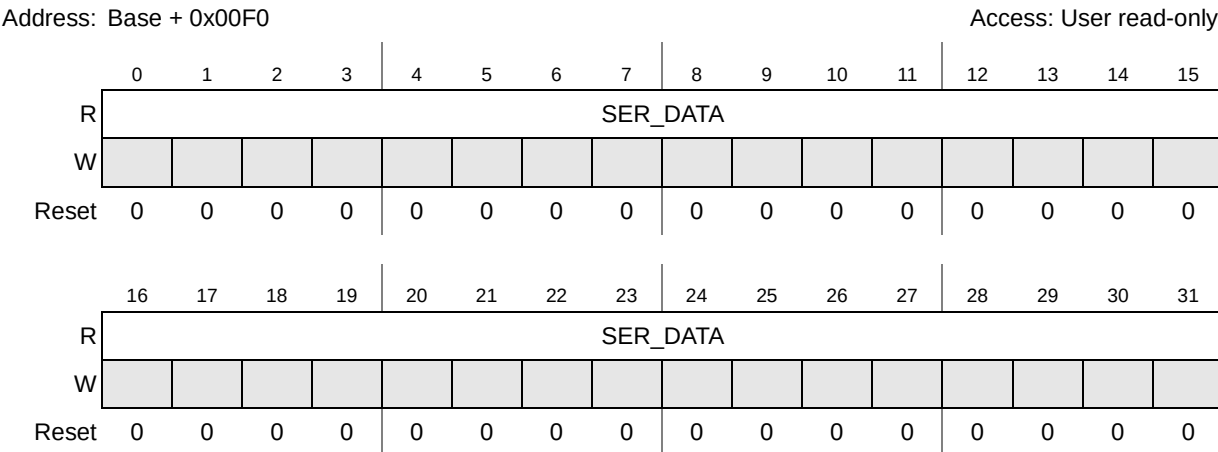


Table 806. DSPIx_SDR1 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the 32 MSB signal states of the 64 parallel input signals. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_SDR1, DSPIx_SDR0} provides the 64-bit data to be serialized.

48.2.10.26 DSPI DSI Alternate Serialization Data Register 1 (DSPIx_ASDR1)

ASDR1 is used by host software to write the 32 MSB of the 64-bit data to be serialized. When the TXSS bit in the DSICR0 is set, the data in the ASDR1 and ASDR0 is the source of the DSI frames. Writes to the ASDR1 take effect on the next frame boundary. The concatenation of {DSPIx_ASDR1, DSPIx_ASDR0} provides the 64-bit alternate serialization data. This register is valid only when DSICR1[DSI64E] is enabled.

Figure 758. DSPI DSI Alternate Serialization Data Register 1 (DSPIx_ASDR1)

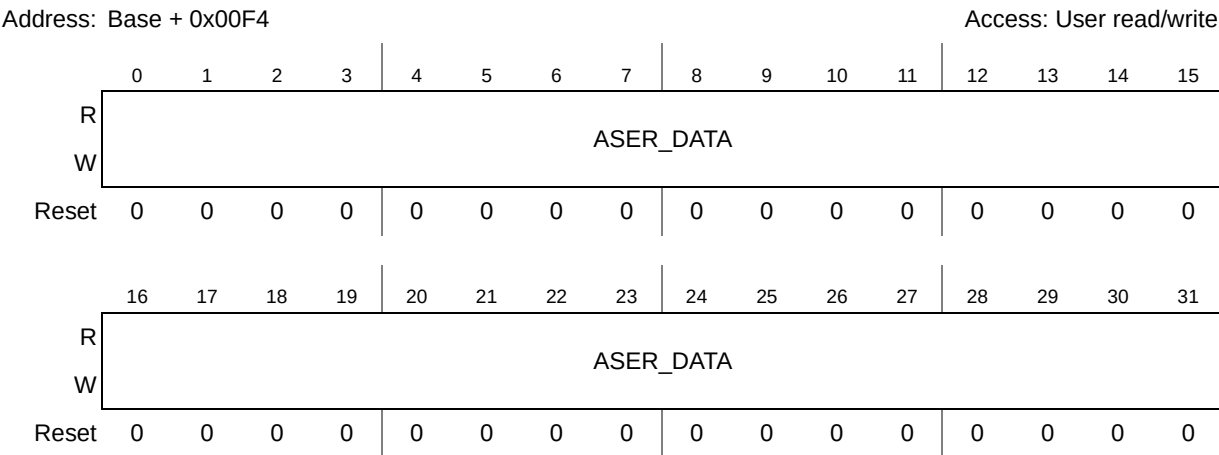


Table 807. DSPIx_ASDR1 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the 32 MSB of the 64-bit alternate data to be serialized. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_ASDR1, DSPIx_ASDR0} provides the 64-bit alternate serialization data.

48.2.10.27 DSPI DSI Transmit Comparison Register 1 (DSPIx_COMPR1)

COMPR1 holds a copy of the 32 MSB of the last transmitted 64-bit DSI data. COMPR1 is read-only. The upper 32 bits of the 64-bit DSI data is transferred to this register as it is loaded into the TX Shift Register. The concatenation of {DSPIx_COMPR1, DSPIx_COMPR0} provides the 64-bit transmit comparison data. This register is valid only when DSICR1[DSI64E] is enabled.

Figure 759. DSPI DSI Transmit Comparison Register 1 (DSPIx_COMPR1)

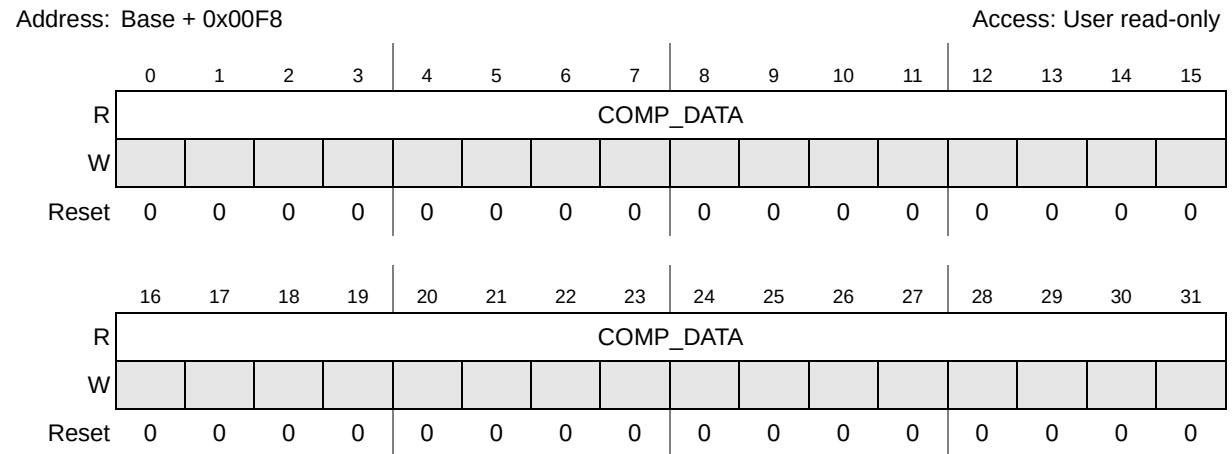


Table 808. DSPIx_COMPR1 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data Holds the 32 MSB of the last serialized 64-bit DSI data. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_COMPR1, DSPIx_COMPR0} provides the last serialized 64-bit DSI data.



48.2.10.28 DSPI DSI Deserialization Data Register 1 (DSPIx_DDR1)

DDR1 holds the 32 MSB signal states for the 64 Parallel Output signals. The DDR1 is read-only and host software can read data from incoming DSI frames.

The concatenation of {DSPIx_DDR1, DSPIx_DDR0} provides the 64-bit Deserialized data. This register is valid only when DSICR1[DSI64E] is enabled.

Figure 760. DSPI DSI Deserialization Data Register 1 (DSPIx_DDR1)

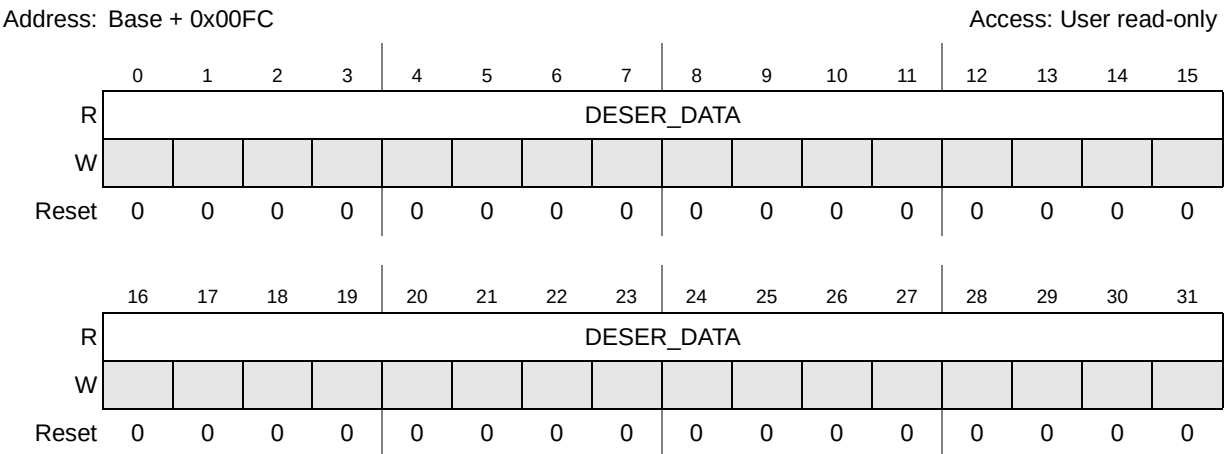


Table 809. DSPIx_DDR1 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data Holds 32 MSB of the deserialized data that is presented as signal states to the 64 Parallel Output signals. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_DDR1, DSPIx_DDR0} provides the 64-bit deserialized data.

48.2.10.29 DSPI DSI Serialization Source Select Register 1 (DSPIx_SSR1)

SSR1 is used to create a combined frame for transmission that contains bits from ASDR1 and from SDR1. Each bit in the SSR1 register selects a corresponding bit to be serialized. When DSICR0[TXSS] is set, the SSR1 register value has no effect. The concatenation of {DSPIx_SSR1, DSPIx_SSR0} provides the 64-bit Serialization Source Select Value. This register is valid only when DSICR1[DSI64E] is enabled.

Figure 761. DSPI DSI Serialization Source Select Register (DSPIx_SSR)

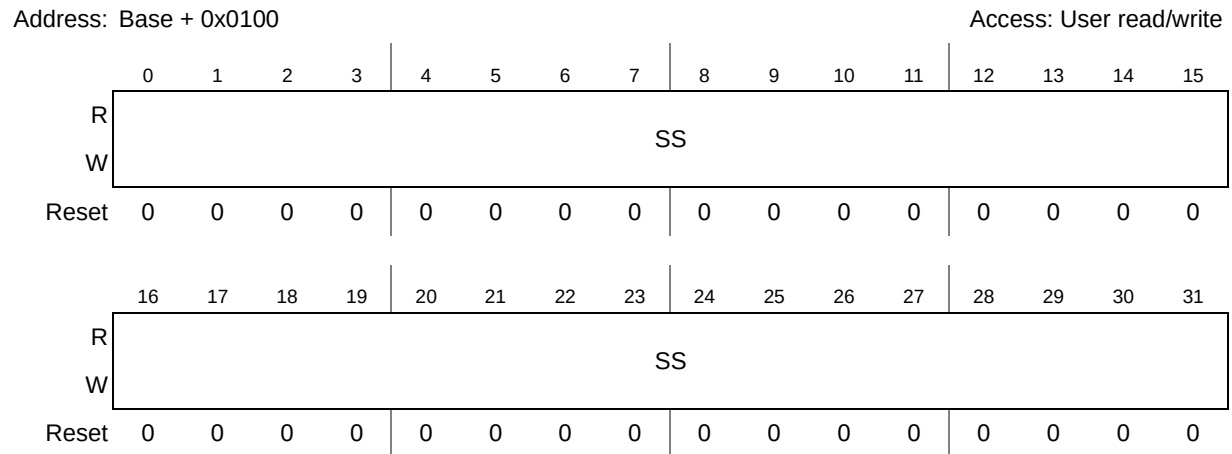


Table 810. DSPIx_SSR field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 MSB of the 64-bit DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_SSR1, DSPIx_SSR0} provides the 64-bit serialization source select value. 0The bit in the transmitted frame is taken from the parallel input pin. 1The bit in the transmitted frame is taken from the ASDR1 register



48.2.10.30 DSPI DSI Parallel Input Select Register 4 (DSPIx_PISR4)

PISR4–7 are used to select each data bit for the transmitted frame from 16 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31 (bit 0 is taken from parallel input 63 when DSICR1[DSI64E] is enabled). When the IPS0 is equal to +1, then bit 0 in the frame is taken from parallel input 1 and so on.

Note that PISR0–7 only preselect the parallel input pins. The final selection to the transmitted frame is made with the SSR0 and SSR1 register bits or the DSICR0[TXSS] bit.

When 64-bit DSI mode is enabled, PISR4-7 Registers are used to handle the extra 32 bits (32 MSB pins of the 64 parallel input pins). If DSICR1[DSI64E] is disabled, then DSPIx_PISR4-7 Registers are invalid.

Figure 762. DSPI DSI Parallel Input Select Register 4 (DSPIx_PISR4)

Address: Base + 0x0104

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPS39				IPS38				IPS37				IPS36			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS35				IPS34				IPS33				IPS32			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 811. DSPIx_PISR4 field descriptions

Field	Description
0–3 IPS39	Input Pin Select 39 Selects the parallel input pin for transmitted frame bit 39.
4–7 IPS38	Input Pin Select 38 Selects the parallel input pin for transmitted frame bit 38.
8–11 IPS37	Input Pin Select 37 Selects the parallel input pin for transmitted frame bit 37.
12–15 IPS36	Input Pin Select 36 Selects the parallel input pin for transmitted frame bit 36.
16–19 IPS35	Input Pin Select 35 Selects the parallel input pin for transmitted frame bit 35.

Table 811. DSPIx_PISR4 field descriptions (continued)

Field	Description
20–23 IPS34	Input Pin Select 34 Selects the parallel input pin for transmitted frame bit 34.
24–27 IPS33	Input Pin Select 33 Selects the parallel input pin for transmitted frame bit 33.
28–31 IPS32	Input Pin Select 32 Selects the parallel input pin for transmitted frame bit 32.

48.2.10.31 DSPI DSI Parallel Input Select Register 5 (DSPIx_PISR5)

Figure 763. DSPI DSI Parallel Input Select Register 5 (DSPIx_PISR5)

Address: Base + 0x0108

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPS47				IPS46				IPS45				IPS44			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS43				IPS42				IPS41				IPS40			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 812. DSPIx_PISR5 field descriptions

Field	Description
0–3 IPS47	Input Pin Select 47 Selects the parallel input pin for transmitted frame bit 47.
4–7 IPS46	Input Pin Select 46 Selects the parallel input pin for transmitted frame bit 46.
8–11 IPS45	Input Pin Select 45 Selects the parallel input pin for transmitted frame bit 45.
12–15 IPS44	Input Pin Select 44 Selects the parallel input pin for transmitted frame bit 44.
16–19 IPS43	Input Pin Select 43 Selects the parallel input pin for transmitted frame bit 43.
20–23 IPS42	Input Pin Select 42 Selects the parallel input pin for transmitted frame bit 42.
24–27 IPS41	Input Pin Select 41 Selects the parallel input pin for transmitted frame bit 41.
28–31 IPS40	Input Pin Select 40 Selects the parallel input pin for transmitted frame bit 40.

48.2.10.32 DSPI DSI Parallel Input Select Register 6 (DSPIx_PISR6)

Figure 764. DSPI DSI Parallel Input Select Register 6 (DSPIx_PISR6)

Address: Base + 0x010C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPS55				IPS54				IPS53				IPS52			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS51				IPS50				IPS49				IPS48			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 813. DSPIx_PISR6 field descriptions

Field	Description
0–3 IPS55	Input Pin Select 55 Selects the parallel input pin for transmitted frame bit 55.
4–7 IPS54	Input Pin Select 54 Selects the parallel input pin for transmitted frame bit 54.
8–11 IPS53	Input Pin Select 53 Selects the parallel input pin for transmitted frame bit 53.
12–15 IPS52	Input Pin Select 52 Selects the parallel input pin for transmitted frame bit 52.
16–19 IPS51	Input Pin Select 51 Selects the parallel input pin for transmitted frame bit 51.
20–23 IPS50	Input Pin Select 50 Selects the parallel input pin for transmitted frame bit 50.
24–27 IPS49	Input Pin Select 49 Selects the parallel input pin for transmitted frame bit 49.
28–31 IPS48	Input Pin Select 48 Selects the parallel input pin for transmitted frame bit 48.

48.2.10.33 DSPI DSI Parallel Input Select Register 7 (DSPIx_PISR7)

Figure 765. DSPI DSI Parallel Input Select Register 7 (DSPIx_PISR7)

Address: Base + 0x0110

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPS63				IPS62				IPS61				IPS60			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS59				IPS58				IPS57				IPS56			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 814. DSPIx_PISR7 field descriptions

Field	Description
0–3 IPS63	Input Pin Select 63 Selects the parallel input pin for transmitted frame bit 63
4–7 IPS62	Input Pin Select 62 Selects the parallel input pin for transmitted frame bit 62
8–11 IPS61	Input Pin Select 61 Selects the parallel input pin for transmitted frame bit 61
12–15 IPS60	Input Pin Select 60 Selects the parallel input pin for transmitted frame bit 60
16–19 IPS59	Input Pin Select 59 Selects the parallel input pin for transmitted frame bit 59
20–23 IPS58	Input Pin Select 58 Selects the parallel input pin for transmitted frame bit 58
24–27 IPS57	Input Pin Select 57 Selects the parallel input pin for transmitted frame bit 57
28–31 IPS56	Input Pin Select 56 Selects the parallel input pin for transmitted frame bit 56

48.2.10.34 DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPIx_DIMR1)

DIMR1 selects bits from the 32 MSB in the received 64-bit DSI frame to be checked to generate the DDI interrupt. The concatenation of {DSPIx_DIMR1, DSPIx_DIMR0} provides the 64-bit Deserialized Data Interrupt Mask Value. This register is valid only when DSICR1[DSI64E] is enabled.

Figure 766. DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPIx_DIMR1)

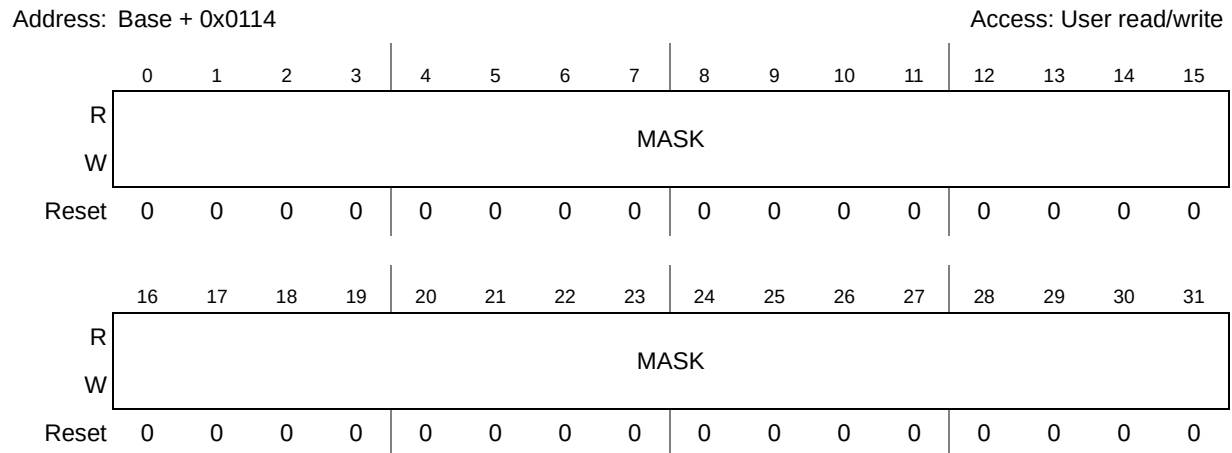


Table 815. DSPIx_DIMR1 field descriptions

Field	Description
0–31 MASK	Mask Defines which of the 32 MSB in the received 64-bit deserialization data should be checked to produce the Deserialized Data Interrupt (DDI). 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.



48.2.10.35 DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPIx_DPIR1)

DPIR1 defines which of the 32 MSB value in the received 64-bit DSI frame generates the DDI interrupt. The concatenation of {DSPIx_DPIR1, DSPIx_DPIR0} provides the 64-bit Deserialized Data Polarity Value. This register is valid only when DSICR1[DSI64E] is enabled.

Figure 767. DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPIx_DPIR1)

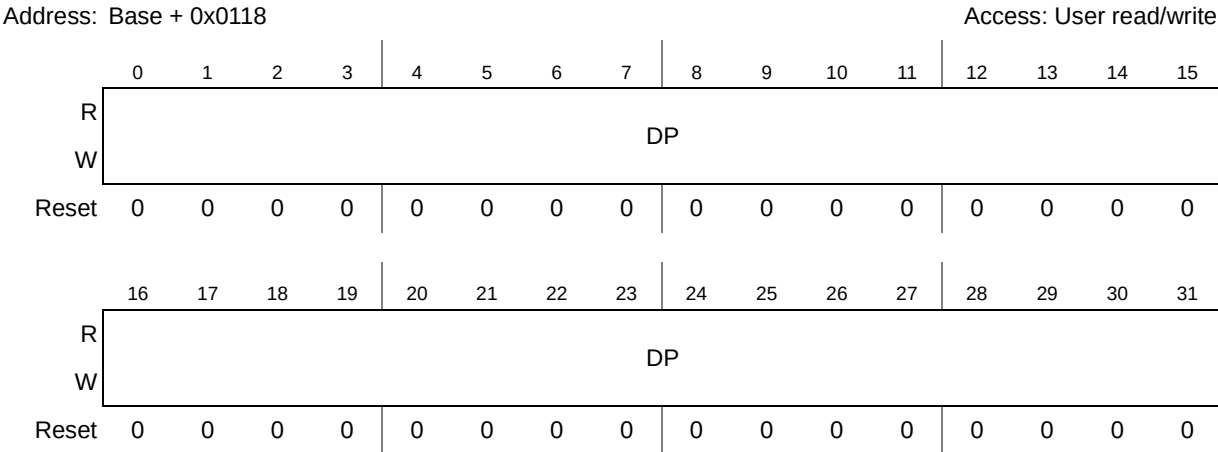


Table 816. DSPIx_DPIR1 field descriptions

Field	Description
0–31 DP	Data Polarity Defines which of the 32 MSB of the received 64-bit deserialization data sets the SR[DDIF] bit. 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

48.2.10.36 DSPI Clock and Transfer Attributes Register Extended (DSPIx_CTAREn)

CTARE registers are used to define the extended transfer attributes for an SPI frame. These registers are valid only when DSPIx_MCR[XSPI] is set.

When the DSPI is configured as an SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used. When the DSPI is configured as an SPI bus slave, the CTARE0 register is used.

Figure 768. DSPI Clock and Transfer Attributes Register Extended (DSPIx_CTAREn)

Address: Base + 0x011C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FMSZE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DTCP										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 817. DSPIx_CTAREn field descriptions

Field	Description
16 FMSZE	Frame Size Extended This field is valid only when DSPIx_MCR[XSPI] is set. This field concatenated with DSPIx_CTAR[FMSZ] defines the frame size of the SPI frames to be transmitted. Effective frame size is the concatenation of {DSPIx_CTAR[FMSZ], DSPIx_CTARE[FMSZE]} plus 1. 0Default Mode. Up to 16-bit SPI frames can be transferred. 1Up to 32-bit SPI frames can be transferred. Each frame transfer will be a result of 2 TX FIFO Pops.
21–31 DTCP	Data Transfer Count Preload This field is valid only when DSPIx_MCR[XSPI] is set. This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame that selected this DSPIx_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

48.2.10.37 DSPI Status Register Extended (DSPIx_SREX)

The DSPIx_SREX contains status fields. The fields reflect the status of the DSPI and indicate the occurrence of events. This register is not writable.

Figure 769. DSPI Status Register Extended (DSPIx_SREX)

Address: Base + 0x013C

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CMDCTR				CMDNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 818. DSPIx_SREX field descriptions

Field	Description
24–27 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
28–31 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

48.3 Register classification for safety

The registers of this module are classified as per safety requirements as follows:

Safety Critical: There are no safety critical registers in this module.

Safety Relevant: The register bits/fields in this category are safety relevant as any random error in these bits will lead to unwanted interference to CPU. The registers containing these fields must be protected by periodic CRC or equivalent check implemented at chip level. The following register bits/fields are classified in this category:

- DSPI Module Configuration Register (DSPIx_MCR)
- DSPI Transfer Count Register (DSPIx_TCR)
- DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPIx_CTARn)
- DSPI Clock and Transfer Attributes Extended Register (In Master Mode) (DSPIx_CTAREN)
- DSPI DMA/Interrupt Request Select and Enable Register (DSPIx_RSER)

- DSPI PUSH FIFO Register In Master Mode (DSPIx_PUSHR) [0:15]
- DSPI DSI Configuration Register 0 (DSPIx_DSICR0)
- DSPI DSI Configuration Register 1 (DSPIx_DSICR1)

Safety Latent: The register bits/fields in this category should be protected by LBIST at chip level. The following register bits/fields are classified in this category:

- DSPI Status Register (DSPIx_SR) [0:15] (for stuck at 0)
- DSPI Status Register Extended (DSPIx_SREX) [0:8] (for stuck at 0)
- DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPIx_DIMR0)
- DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPIx_DIMR1)
- DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPIx_DPIR0)
- DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPIx_DPIR1)

Non-Safety: All other registers and bits fall in this category.

48.4 Functional description

The Deserial Serial Peripheral Interface (DSPI) block supports full-duplex, synchronous serial communications between MCUs and peripheral devices. The DSPI can also be used to reduce the number of pins required for I/O by serializing and deserializing up to 64 parallel input/output signals. All communications are done with SPI-like protocol.

The DSPI has the following configurations:

- SPI Configuration in which the DSPI operates as a basic SPI or a queued SPI.
- DSI Configuration in which the DSPI serializes and deserializes Parallel Input/Output signals or bits from memory-mapped register.
- CSI Configuration in which the DSPI combines the functionality of the SPI and DSI configurations.

The DCONF field in the DSPI Module Configuration Register (MCR) determines the DSPI Configuration. See [Section 48.2.10.1, DSPI Module Configuration Register \(DSPIx_MCR\)](#) for the DSPI configuration values.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command. The Extended SPI Mode (DSPIx_MCR[XSPI]) further allows the usage of CTAREn (CTARn Extended) registers which allows the user to send multiple Data frames (of up to 32-bit frame size) using a single Command frame.

The DSI configuration statically selects which CTAR to use. In CSI, Configuration priority logic determines if SPI data or DSI data is transferred and dictates what CTAR is used for the data transfer.

See DSPI Clock and Transfer Attributes Registers for information on the fields of the CTARs.

See DSPI Clock and Transfer Attributes Registers Extended for information on the fields of the CTARE registers.

Typical master to slave connections are shown in [Figure 770](#). When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The

data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the TCF bit in the SR is set to indicate a completed transfer.

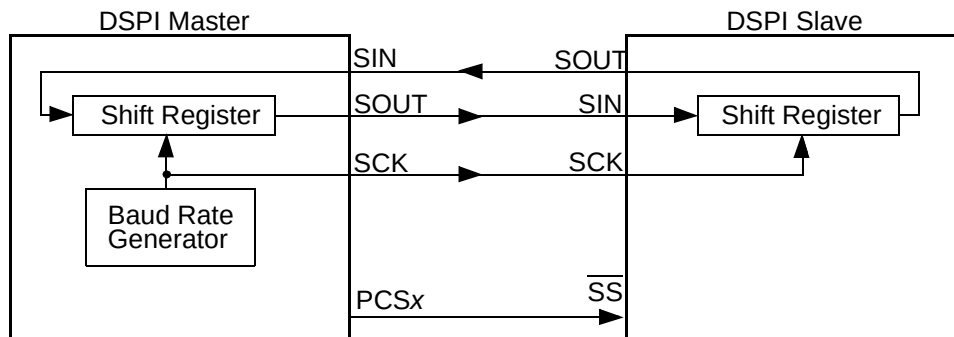


Figure 770. SPI serial protocol overview

Generally more than one slave device can be connected to the DSPI master.

Eight Peripheral Chip Select (PCS) signals of the DSPI masters can be used to select which of the slaves to communicate with. Refer to the chip configuration chapter for the number of PCS signals used in this MCU.

The three DSPI configurations share transfer protocol and timing properties which are described independently of the configuration in [Section 48.4.6, Transfer formats](#). The transfer rate and delay settings are described in [Section 48.4.5, DSPI baud rate and clock delay generation](#).

48.4.1 Start and Stop of DSPI transfers

The DSPI has two operating states: STOPPED and RUNNING. The states are independent of DSPI configuration. The default state of the DSPI is STOPPED. In the STOPPED state no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results. In the RUNNING state serial transfers take place.

The TXRXS bit in the SR indicates what state the DSPI in. The bit is set if the module in RUNNING state.

The DSPI is started (DSPI transitions to RUNNING) when all of the following conditions are true:

- SR[EOQF] bit is clear
- MCU is not in the debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The DSPI stops (transitions from RUNNING to STOPPED) after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- MCU in the debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

48.4.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The DSPI is in SPI Configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to DSPI RAM queues to a transmit FIFO (TX FIFO) buffer. The received data is stored in entries in the Receive FIFO (RX FIFO) buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the DSPI. The FIFO buffers operation is described in [Section 48.4.2.4, Transmit First In First Out \(TX FIFO\) buffering mechanism](#), [Section 48.4.2.5, Command First In First Out \(CMD FIFO\) buffering mechanism](#) and [Section 48.4.2.6, Receive First In First Out \(RX FIFO\) Buffering Mechanism](#). The interrupt and DMA request conditions are described in [Section 48.4.12, Interrupts/DMA requests](#).

The SPI Configuration supports two block-specific modes —master mode and slave mode. In master mode, the DSPI initiates and controls the transfer according to the fields of the executing CMD FIFO entry. In slave mode, the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field space is reserved.

48.4.2.1 Master Mode

In SPI master mode the DSPI initiates the serial transfers by controlling the Serial Communications Clock (SCK) and the Peripheral Chip Select (PCS) signals. The executing SPI CMD FIFO entry determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See DSPI PUSH FIFO Register (PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI master mode, each SPI frame to be transmitted has a command associated with it allowing for transfer attribute control on a frame by frame basis. In Extended SPI Master Mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. In addition, the Extended SPI Mode allows for larger frame sizes of up to 32 bits.

48.4.2.2 Slave Mode

In SPI slave mode the DSPI responds to transfers initiated by a SPI bus master. The DSPI does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase and frame size must be set for successful communication with a SPI master. The SPI slave mode transfer attributes are set in the CTAR0 and CTARE0. The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

48.4.2.3 FIFO Disable Operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately; setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO Disable mechanisms are transparent to the user and to host software; Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO is disabled (CMD FIFO is also disabled), the TFFF, TFUF, CMDFFF, CMDCTR and TXCTR fields in SR, SREX behave as if there is a one-entry FIFO but the contents of the TXFR registers, TXNXTPTR and CMDNXTPTR are undefined. Likewise, when the RX FIFO is disabled, the RFDF, RFOF and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

48.4.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 4 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data field of DSPI PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time an 8 or 16-bit write takes place to the Data field of DSPI_PUSH or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates which TX FIFO Entry will be transmitted during the next transfer. The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO are used separately. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

48.4.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSH. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSH is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See [Section 48.4.12.2, Transmit FIFO Fill interrupt or DMA request](#) for details.

The DSPI ignores attempts to push data to a full TX FIFO, the state of the TX FIFO does not change and no error condition is indicated.

48.4.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled, if the frame size of SPI Data to be transmitted is more than 16 bits, then it causes two Data entries to be popped from TX FIFO simultaneously which are transferred to the shift register. The first of the two popped entries forms the 16 LSB bits of the SPI frame. Such an operation also causes TX FIFO Counter to decrement by two. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a DSPI slave while the slave's DSPI TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See [Section 48.4.12.9, Transmit FIFO Underflow interrupt request](#) for details.

48.4.2.5 Command First In First Out (CMD FIFO) buffering mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. When Extended SPI Mode (DSPIx_MCR[XSPI]) is disabled, the TX FIFO and CMD FIFO must be filled together. When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled, the TX FIFO and CMD FIFO can be filled independently.

The CMD FIFO holds four entries, each representing command fields. The number of entries in the CMD FIFO is device-specific. SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (DSPIx_MCR[XSPI]) is disabled, every CMD FIFO entry has a corresponding single TXFIFO entry attached to it.

When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled, every CMD FIFO entry can have multiple TXFIFO entries attached to it. Thus a single CMD FIFO entry can be used to transmit multiple TX FIFO entries. The CTARE[DTCP] field decides the number of SPI Data Frames having frame size as {FMSIZE, FMSZ} to be transmitted using the current Command Entry. The CTAR/CTARE registers pointed by the CTAS field in the Command frame gives FMSZ and FMSIZE fields respectively. The time for which a command entry is in use is known as a Command Cycle. The Busy Flag SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO Counter field (CMDCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the CMD FIFO. The TXCTR is updated every time an 8 or 16-bit write takes place on the lower half of DSPI_PUSH or SPI data is transferred into the shift register from the TX FIFO.

The CMDNXTPTR field indicates which CMD FIFO Entry will be used during the next command cycle. The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO are used separately. The CMDNXTPTR field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

48.4.2.6 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds four received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the DSPI POP FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR register number and it rolls over after reaching the maximum.

48.4.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time a SPI frame is transferred to the RX FIFO the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

48.4.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the DSPI POP FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a '1' to it.

48.4.3 Deserial Serial Interface (DSI) configuration

The DSI Configuration supports pin count reduction by serializing parallel input signals or register bits and shifting them out in a SPI-like protocol. The timing and transfer protocol is described in [Section 48.4.6, Transfer formats](#). The received serial frames are converted to a parallel form (deserialized) and placed on the Parallel Output signals or in the DSPI_DDR. The various features of the DSI Configuration are set in DSPI DSI Configuration Registers (DSPI_DSICR1/0).

The DSI frames can be from 4 to 32 bits. This can go up to 64 bits when DSICR1[DSI64E] is enabled. With Multiple Transfer Operation (MTO) the DSPI supports serial chaining of DSPI blocks within a device to create DSI frames up to 64 bits, consisting of concatenated bits from multiple DSPIs. The DSPI also supports parallel chaining allowing several DSPIs and off-chip SPI devices to share the same Serial Communications Clock (SCK) and Peripheral Chip Select (PCS) signals. See [Section 48.4.3.6, Multiple Transfer Operation \(MTO\)](#) for details on the serial and parallel chaining support.

Multiple Transfer Operation (MTO) and DSICR1[DSI64E] are not supported together and cannot be enabled simultaneously.

48.4.3.1 DSI master mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has four different conditions that can initiate a transfer:

- Continuous
- Change in data
- Trigger signal
- Trigger signal combined with a change in data

The four transfer initiation conditions are described in [Section 48.4.3.5, DSI transfer initiation control](#). Transfer attributes are set during initialization. DSICR0[DSICTAS] field determines which of the DSPI_CTARs will control the transfer attributes.

48.4.3.2 Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by a SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers. Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI_CTAR1. The data is shifted out with MSB first.

If DSICR0[CID] is set and the data in the DSPI_COMPR1/0 differs from the selected source of the serialized data, the slave DSPI will assert the MTRIG signal. If the slave's HT signal is asserted and the TRRE is set, the slave DSPI asserts MTRIG. These features are included to support chaining of several DSPI. Details about the MTRIG signal can be found in [Section 48.4.3.6, Multiple Transfer Operation \(MTO\)](#).

48.4.3.3 DSI serialization

In the DSI configuration from 4 to 16 bits can be serialized using two different sources. DSICR0[TXSS] bit selects between the DSPI DSI Serialization Data Register (DSPI_SDR1/0) and the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR1/0) as the source of the serialized data. The DSPI_SDR0 holds the latest 32 parallel input signal values which is sampled at every rising edge of the protocol clock. When DSICR1[DSI64E] is enabled, 64-bit DSI Frames are supported and hence DSPI_SDR1 holds the 32 MSB parallel input signal values which is sampled at every rising edge of the protocol clock. The DSPI_ASDR1/0 registers are written by host software and used as an alternate source of serialized data. (DSPI_ASDR1 is only usable when DSICR1[DSI64E] is enabled).

The DSPI has to sample the (SDR/ASDR) inputs before it transmits a DSI frame. Due to the asynchronous Clock structure within DSPI, the sampling of inputs come under Register Clock domain whereas DSI frame transmission logic lies in Protocol Clock domain. Hence the sampled data (from ASDR1/0 or SDR1/0) must be synchronized into the Protocol Domain before being loaded into the Shift Register for transmission.

To prevent Clock Domain clocking issues, SDR/ASDR inputs should not change while they are being sampled into the Protocol Domain. This is ensured in the following manner for the ASDR inputs. The initial write performed on ASDR register will cause the 32-bit ASDR input to be latched into a temporary register which is then locked (further latching is prevented) until this data gets synchronized into the Protocol Domain. Once synchronized, the lock is removed and the next ASDR write will again trigger this process. If there was a write performed on ASDR while the lock was asserted, this will be taken into consideration as soon as the previous synchronization process completes and lock gets de-asserted.

In case, multiple writes were performed on ASDR when lock was asserted after the first write, only the latest value will be considered for synchronization once the lock gets de-asserted. The above locking scheme and the synchronization procedure is true for both ASDR0 and ASDR1 registers. The entire process explained above is also true for the SDR0 and SDR1 inputs, the only difference being that SDR inputs are delayed internally by a single clock to detect a change in the SDR input pins.

Note: *If SDR0 and SDR1 inputs are both changed at the same time instant, then the synchronization logic ensures that both SDR0 and SDR1 contribute to the same DSI frame.*

Once the synchronization process for a particular Deserial input begins, it takes up to five Register clock periods and five Protocol clock periods before the synchronization process for the next Deserial input can begin. Hence the writes performed on ASDR0/1 and SDR0/1 are constrained with this timing.

The DSPI_PISR0–3 registers (DSPI_PISR0–7 registers if DSICR1[DSI64E] is enabled) allow to change the relative position of the Parallel input pins in the transmitted frame. Each transmitted frame bit can be selected from 16 adjacent parallel inputs by writing IPS_n fields. The IPS_n field is treated as a 4-bit integer number, representing numbers from -8 to 7. The parallel input pin number, selected by IPS_n field is defined by the sum of the IPS_n field number (*n*) and the IPS_n field value. If the operation result is negative the number 32 should be added. If the result is higher than 32, 32 should be subtracted from the result. (When DSICR1[DSI64E] is enabled, if the above operation result is negative, the number 64 should be added. If the result is higher than 64, 64 should be subtracted from the result).

For example, IPS0, set to minus 1 (binary 1111), preselects parallel input 31 to 0 position in the transmitted frame. (This preselects parallel input 63 to 0 position in the transmitted frame when DSICR1[DSI64E] is set).

IPS6, set to 3 (binary 0011), preselects parallel input 9 to be bit number 6 in the transmitted frame, while the value minus 2 (1110) preselects parallel input 4.

IPS31, set to minus 8 (binary 1000), preselects parallel input 23 to be bit number 31 in the transmitted frame.

The parallel input pin state, to be transmitted, should be selected by TXSS or DSPI_SSR1/0 registers and the frame size should be higher than the bit position in the preselected frame.

DSPI_SSR1/0 registers provide additional way to create the frame for transmission. Each bit from this register is OR'd with the TXSS bit and controls individual transmitted bit source. This way, the transmitted frame can have any combination of the DSPI_SDR1/0 and DSPI_AS DR1/0 bits. This feature allows control SPI based devices, requiring control and data fields in the frame. The control field may come from DSPI_AS DR1/0 registers, set by the CPU, while data field can be generated by device peripheral modules, like PWM timers.

A copy of the last DSI frame shifted out of the Shift Register is stored in the DSPI DSI Transmit Comparison Registers (DSPI_COMP R1/0). This register provides added visibility for debugging and it serves as a reference for transfer initiation control. [Figure 771](#) shows the DSI serialization logic.

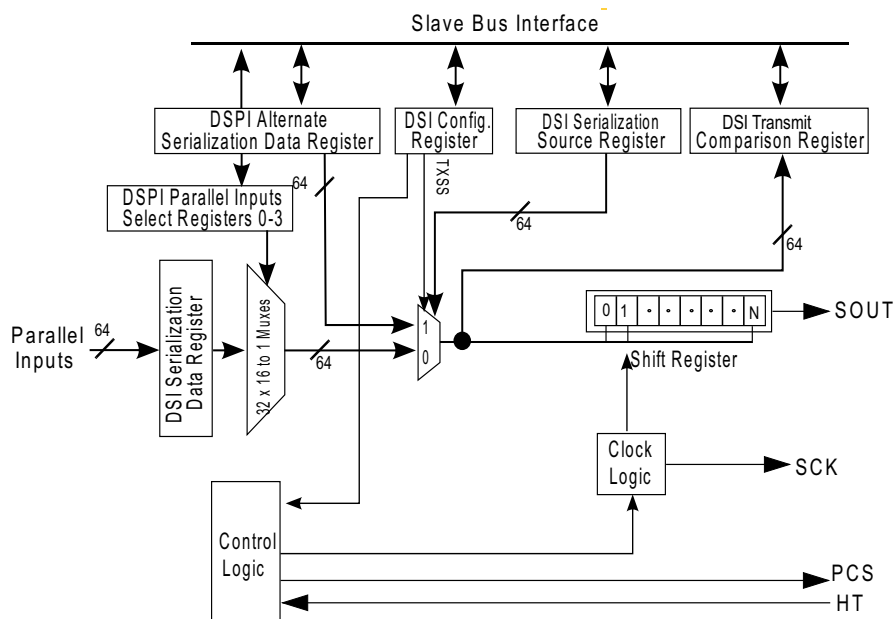


Figure 771. DSI serialization diagram

48.4.3.4 DSI deserialization

When all bits in a DSI frame have been shifted in, the frame is copied to the DSPI DSI Deserialization Data Register (DSPI_DDR1/0). This register presents the deserialized data as Parallel Output signal values. The DSPI_DDR1/0 is memory-mapped to allow host software to read the deserialized data directly.

The received data is bit-wise compared to the value of the DSI Deserialized Data Polarity Interrupt Register (DSPI_DPIR1/0), bit-wise AND'ed with DSI Deserialized Interrupt Mask Register (DSPI_DIMR1/0) and the results OR'ed to produce DDIF flag in the DSPI_SR. Which in turn can cause DDIF interrupt request or DMA request based on RSER[DDIF_RE] and RSER[DDIF_DIRS] bits and/or stop DSI frame transmissions if DSICR0[DMS] bit is set. [Figure 772](#) shows the DSI Deserialization logic.

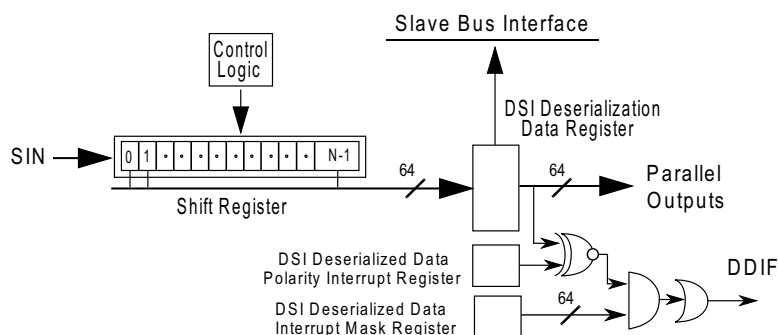


Figure 772. DSI deserialization diagram

48.4.3.5 DSI transfer initiation control

Data transfers for a master DSPI in DSI configuration are initiated by a condition. The transfer initiation conditions are selected by the TRRE and CID bits in the DSPI_DSICR0 register. [Table 819](#) lists the four transfer initiation conditions.

Table 819. DSI data transfer initiation control

DSPI_DSICR0 bits		Transfer initiation control
TRRE	CID	
0	0	Continuous
0	1	Change in Data
1	0	Triggered
1	1	Triggered or Change in Data

48.4.3.5.1 Continuous control

For continuous control, a new DSI frame shifts out when the previous transfer cycle has completed and the delay after transfer (t_{DT}) has elapsed.

48.4.3.5.2 Change in data control

For change in data control, a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in the DSPI_COMPR1/0. When the data, selected for the transfer from the DSPI_SDR1/0 and DSPI_AS DR1/0 registers is different from the data in the DSPI_COMPR1/0 a new DSI frame is transmitted. The MTRIG output signal is asserted every time a change in data is detected.

48.4.3.5.3 Triggered control

For Triggered Control, initiation of a transfer is controlled by the Hardware Trigger signal (HT). DSICR0[TPOL] bit selects the active edge of HT. For HT to have any affect, DSICR0[TRRE] bit must be set.

48.4.3.5.4 Triggered or change in data control

For triggered or change in data control initiation of a transfer is controlled by the HT signal or by the detection of a change in data to be serialized.

It is not advisable to configure a DSPI Slave with the combination DSICR0[DCONT] = 1, DSICR0[CID] = 1; since the following statement then holds true.

Once slave select is asserted, there is a very small window (of up to three module clocks), after every SOF (Start of Frame) during which if a write on AS DR/SDR registers occur, the next frame to be transmitted would be loaded into the shift register. However it is not feasible that such a write is initiated and completed within this short span of time; hence the data transmitted by DSPI slave would lag by one data frame.

48.4.3.6 Multiple Transfer Operation (MTO)

In DSI configuration, the MTO feature allows for multiple DSPIs within a device to be chained together in a parallel or serial configuration. The parallel chaining allows multiple DSPIs internal to a device and multiple SPI devices external to a device to share SCK and

PCS signals, thereby saving the device pins. The serial chaining allows bits from multiple DSPIs to be concatenated into a single DSI frame. MTO is enabled by setting DSICR0[MTOE] bit.

In parallel and serial chaining there is one bus master and multiple bus slaves. The bus master initiates and controls the transfers, but the DSPI slaves generate trigger signals for the bus DSPI master when an internal condition in the slave warrants a transfer. The DSPI slaves also propagate triggers from other slaves to the master. When a DSPI slave detects a trigger signal on its HT input, the slave generates a trigger signal on the MTRIG output. Serial and parallel chaining require multiplexing of signals external to the DSPI.

TSB operation is not available in MTO mode. DSICR0[TSBC] and DSICR0[MTOE] bits should not be set simultaneously.

DSI64 mode is also not available in MTO Mode. DSICR1[DSI64E] and DSICR0[MTOE] bits must not be set simultaneously.

48.4.3.6.1 Parallel chaining

Parallel chaining allows multiple DSPIs internal to a device and multiple SPI/DSI devices external to a device to share common SCK and PCS signals thereby saving pins. Two pins are saved per pair of DSPI/SPI. [Figure 773](#) shows an example of how the blocks can be connected in a device.

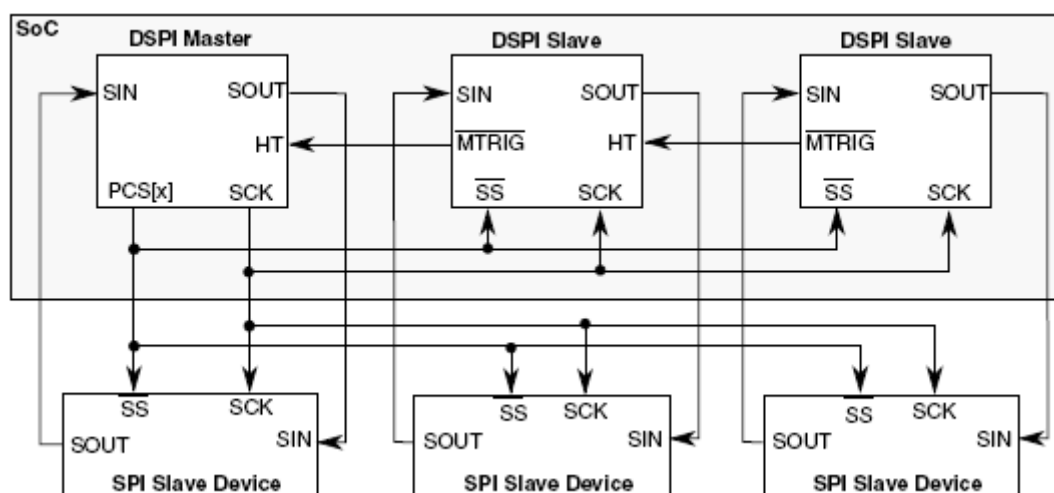


Figure 773. Example of parallel chaining of DSPIs

The DSPI master controls and initiates all transfer, but the DSPI slaves have a trigger output signal MTRIG that indicates to the master DSPI to start a transfer. When the DSPI slave has a change in its data to be serialized, it generates a pulse on the MTRIG signal to the master DSPI, which initiates the transfer. When a DSPI slave has its HT signal asserted it also generates a pulse on its MTRIG signal thereby propagating trigger signals from other DSPI slaves to the DSPI master.

To enable MTRIG signals generation the MTO has to be enabled for the DSPI slaves. The chained DSPI modules also should be in DSI configuration. The DSPI_DSICR0[MTOCNT] field must be written with the same number of bits to be transferred for each slave DSPI.

The MTOCNT value must also match the FMSZ field of the DSPI_CTAR1 in the DSPI slaves and selected DSPI_CTAR for the master DSPI module.

48.4.3.6.2 Serial chaining

The serial chaining allows transfers of DSI frames of up to a total of 64 bits, using transfers of smaller DSI frames concatenated together by multiple DSPIs. [Figure 774](#) shows an example of how the blocks can be connected in a device.

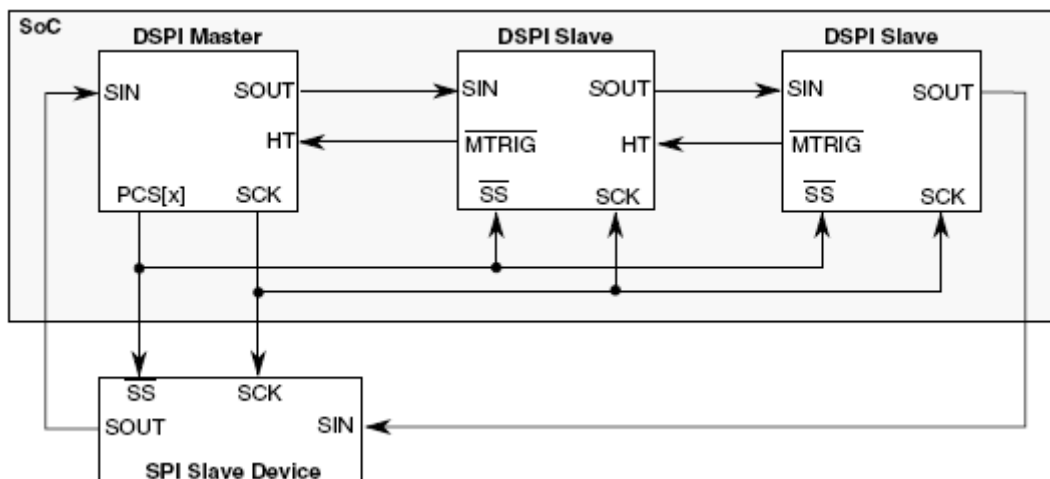


Figure 774. Example of serial chaining of DSPIs

The SOUT of the DSPI master is connected to the SIN of the first DSPI slave. The SOUT of the first DSPI slave is connected to the SIN input of the second slave and so on. The SOUT of the last DSPI slave is connected to the SIN of the external SPI slave. The SOUT of the external SPI slave is connected to the SIN of the DSPI master.

The DSPI master controls and initiates all transfers, but the slave DSPIs use the trigger output signal MTRIG to indicate to the DSPI master that a trigger condition has occurred. When a DSPI slave has a change in data to be serialized it asserts the MTRIG signal to the DSPI master which initiates the transfer. When a DSPI slave has its HT signal asserted it will assert its MTRIG signal thereby propagating trigger signals from other DSPI slaves to the DSPI master.

The DSPI_DSICR0[MTOE] bit must be set in master and all slave devices. The DSPI_DSICR0[MTOCNT] field for the master and all slave devices must be written with the same value. The value must equal the sum of all FMSZ fields in the selected DSPI_CTARs for the DSPI master and all DSPI slaves. For example if one 16-bit DSI frame is created by concatenating eight bits from the DSPI master, and four bits from each of the DSPI slaves in the above figure, the DSPI master frame size must be set to eight in the FMSZ field, and the DSPI slaves frame size must be set to four. The largest DSI frame supported by the MTOCNT field is 64 bits. Any number of DSPIs can be connected together to concatenate DSI frames, as long as each DSPI transfers a minimum of 4 bits and a maximum of 32 bits and the total size of the concatenated frame is less than 65 bits long.

48.4.4 Combined Serial Interface (CSI) configuration

The CSI Configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis. CSI Configuration allows interleaving of DSI data frames from the parallel input signals with SPI data frames from the TX FIFO. The data returned from the bus slave is either used to drive the Parallel Output signals or it is stored in the RX FIFO. The CSI Configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI Configuration when the DCONF field in the DSPI_MCR is 0b10. [Figure 775](#) shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

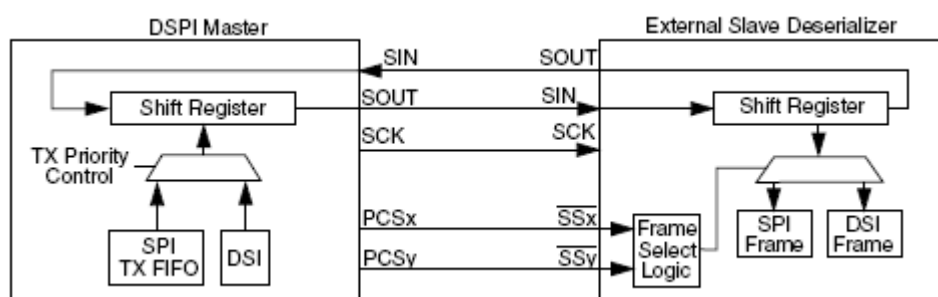


Figure 775. Example of System using DSPI in CSI Configuration

In CSI Configuration the DSPI transfers DSI data based on [Section 48.4.3.5, DSI transfer initiation control](#). When there are SPI commands in the TX FIFO, the SPI data has priority over the DSI frames. When the TX FIFO or CMD FIFO is empty, DSI transfer resumes.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTARs associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy. The CSI Configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the Parallel Output signals. Data returned from the external slave while a SPI frame is transferred is moved to the RX FIFO. The TX FIFO, CMD FIFO and RX FIFO are fully functional in CSI mode.

48.4.4.1 CSI serialization

Serialization in the CSI configuration is similar to serialization in DSI Configuration. The transfer attributes for SPI frames are determined by the DSPI_CTAR selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI_CTAR selected by the DSICR0[DSICTAS] field.

The parallel inputs signal states are latched into the DSPI DSI Serialization Data Register (DSPI_SDR) on the rising edge of every protocol clock and serialized based on the transfer initiation control settings in the DSPI_DSICR1/0. When SPI frames are written to the TX FIFO (and when CMD FIFO is not empty) they have priority over DSI data and are transferred at the next frame boundary. A copy of the most recently transferred DSI frame is stored in the DSPI_COMPR1/0. The Transfer Priority Logic selects the source of the serialized data and asserts the appropriate PCS signal.

Separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI frames.

48.4.4.2 CSI deserialization

The deserialized frames in CSI Configuration goes into the DSPI_DDR1/0 or the RX FIFO based on the transfer priority logic. When DSI frames are transferred the returned frames are deserialized and latched into the DSPI_DDR1/0. When SPI frames are transferred the returned frames are deserialized and written to the RX FIFO.

48.4.5 DSPI baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. [Figure 776](#) shows conceptually how the SCK signal is generated.

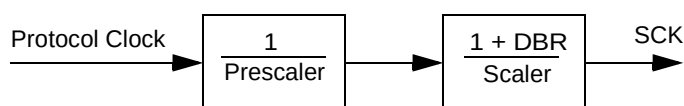


Figure 776. Communications clock prescalers and scalers

48.4.5.1 Baud rate generator

The baud Rate is the frequency of the Serial Communication Clock (SCK). The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. [Table 820](#) shows an example of how to compute the baud rate.

Table 820. Baud rate computation example

f_p	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

Note: The clock frequencies mentioned in [Table 820](#) are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

48.4.5.2 PCS to SCK delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See [Table 821](#) for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTARx registers select the PCS to SCK delay by the formula in the CSSCK field description. [Table 821](#) shows an example of how to compute the PCS to SCK delay.

Table 821. PCS to SCK delay computation example

f_p	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK delay
100 MHz	0b01	3	0b0100	32	0.96 μ s

PCSCSK and CSSCK fields have no effect in TSB configuration.

Note: The clock frequency mentioned in [Table 821](#) is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

48.4.5.3 After SCK delay (t_{ASC})

The After SCK delay is the length of time between the last edge of SCK and the negation of PCS. See [Table 822](#) for an illustration of the After SCK delay. The PASC and ASC fields in the CTARx registers select the After SCK Delay by the formula in the ASC field description. [Table 822](#) shows an example of how to compute the After SCK delay.

Table 822. After SCK delay computation example

f_p	PASC	Prescaler	ASC	Scaler	After SCK delay
100 MHz	0b01	3	0b0100	32	0.96 μ s

PCASC and ASC fields have no effect in TSB configuration.

Note: The clock frequency mentioned in [Table 822](#) is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

48.4.5.4 Delay after transfer (t_{DT})

The delay after transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See [Table 823](#) for an illustration of the delay after transfer. the pdt and dt fields in the CTARx registers select the Delay after Transfer by the formula in the DT field description. [Table 823](#) shows an example of how to compute the delay after transfer.

Table 823. Delay after transfer computation example

f_p	PDT	Prescaler	DT	Scaler	Delay after transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

Note: The clock frequency mentioned in [Table 823](#) is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in non-continuous clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] bitfield description. When in continuous clock mode and TSB is not enabled, the delay is fixed at 1 SCK period.

In TSB mode the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. See detailed information on [Section 48.4.9, Timed Serial Bus \(TSB\)](#).

48.4.5.5 Peripheral Chip Select Strobe Enable ($\overline{PCSS}$)

The PCSS signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the DSPI is in master mode and the PCSSE bit is set in the MCR, PCSS provides a signal for an external demultiplexer to decode the PCS[0]–PCS[4] and PCS[6]–PCS[7] signals into as many as 128 glitch-free PCS signals. [Figure 777](#) shows the timing of the PCSS signal relative to PCS signals.

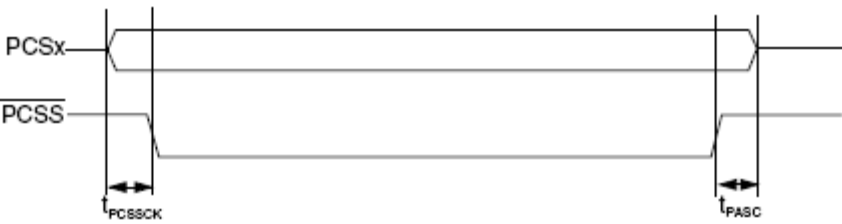


Figure 777. Peripheral Chip Select Strobe Timing

The delay between the assertion of the PCS signals and the assertion of PCSS is selected by the PCSSCK field in the CTAR based on the following formula:

Equation 29
$$t_{PCSSCK} = \frac{1}{f_P} \times PCSSCK$$

At the end of the transfer the delay between $\overline{PCSS}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

Equation 30
$$t_{PASC} = \frac{1}{f_P} \times PASC$$

[Table 824](#) shows an example of how to compute the t_{PCSSCK} delay.

Table 824. Peripheral Chip Select Strobe Assert Computation Example

f_P	PCSSCK	Prescaler	Delay before transfer
100 MHz	0b11	7	70.0 ns

[Table 825](#) shows an example of how to compute the t_{PASC} delay.

Table 825. Peripheral Chip Select Strobe Negate Computation Example

f_P	PASC	Prescaler	Delay after transfer
100 MHz	0b11	7	70.0 ns

The $\overline{PCSS}$ signal is not supported when Continuous Serial Communication SCK or TSB mode are enabled.

Note: The clock frequency mentioned in the preceding [Table 824](#) and [Table 825](#) is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

48.4.6 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL—Selects the idle state polarity of the SCK
- CPHA—Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in slave mode these values must be identical to the master device settings to ensure proper transmission. In SPI slave mode, only CTAR0 is used.

In DSI slave mode, only CTAR1 is used.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0
- Classic SPI with CPHA = 1
- Modified Transfer format with CPHA = 0
- Modified Transfer format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI and DSI Configurations, the DSPI provides the option of keeping the PCS signals asserted between frames. See [Section 48.4.6.5, Continuous Selection Format](#) for details.

48.4.6.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in [Figure 778](#) is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

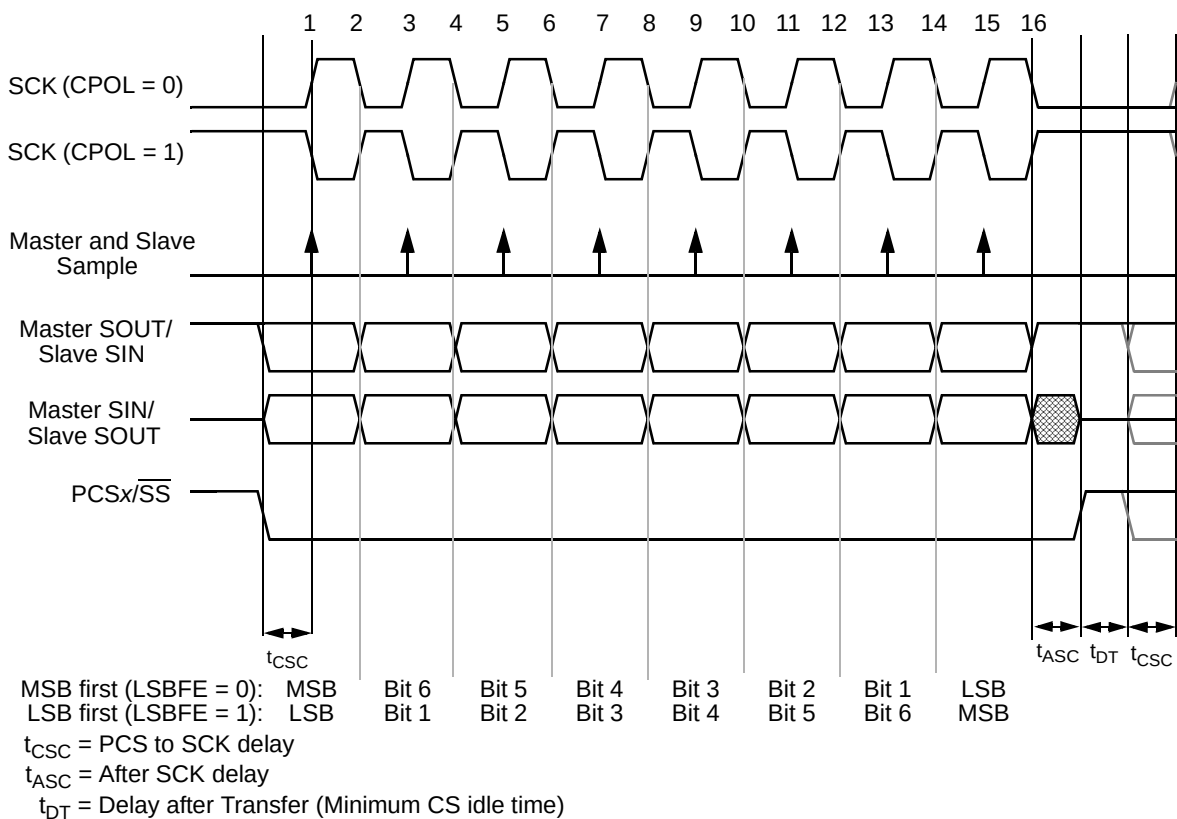


Figure 778. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

48.4.6.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in [Figure 779](#) is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges

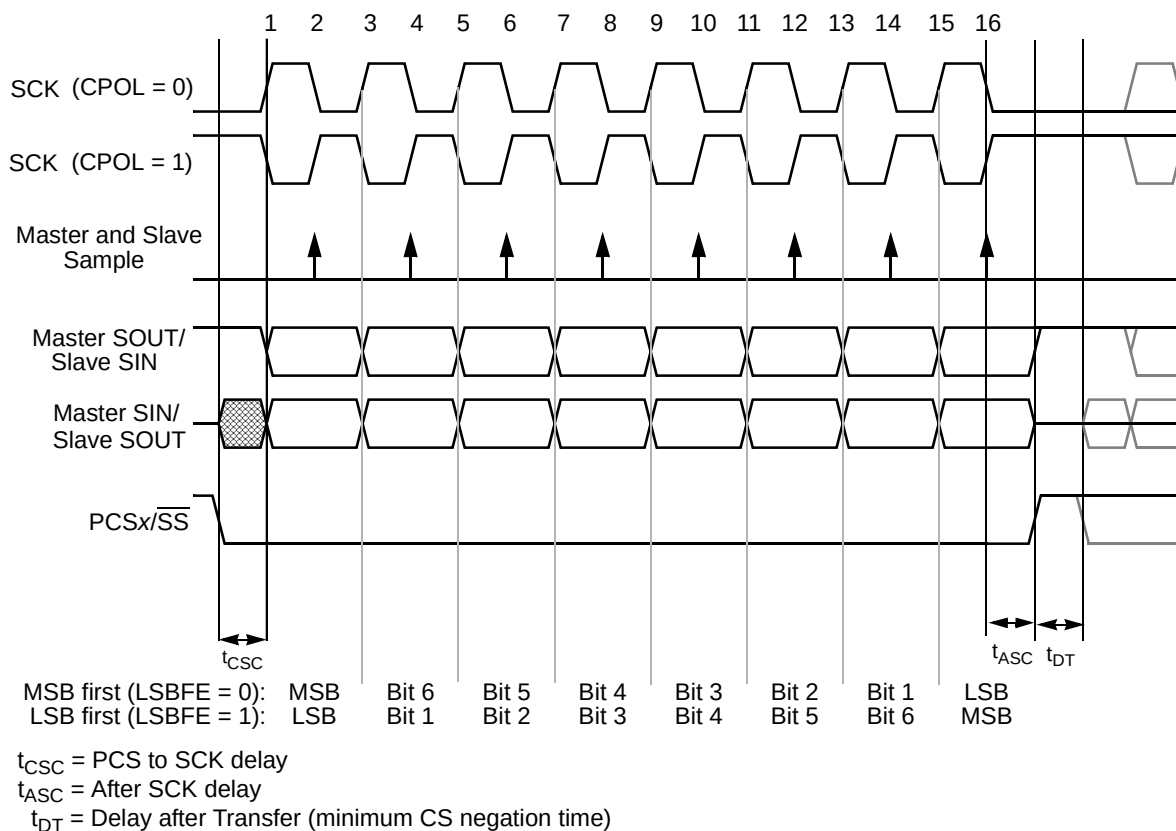


Figure 779. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

48.4.6.3 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actually sample point is delayed by 1 protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE = 1. Timing delays shown are:

- t_{CSC} – PCS to SCK assertion delay
- t_{ACS} – After SCK PCS negation delay
- t_{su_ms} – master SIN setup time
- t_{hd_ms} – master SIN hold time
- t_{vd_sl} – slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- t_{su_sl} – data setup time on slave data input
- t_{hd_sl} – data hold time on slave data input
- t_p – protocol clock period

Figure 780 shows the modified transfer format for CPHA = 0 and $f_p/f_{sck} = 4$. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behaviors are shown.

- Signal, marked “SOUT of Ext Slave”, presents regular SPI slave serial output.
- Signal, marked “SOUT of DSPI Slave”, presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note: In the following diagrams, f_p represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

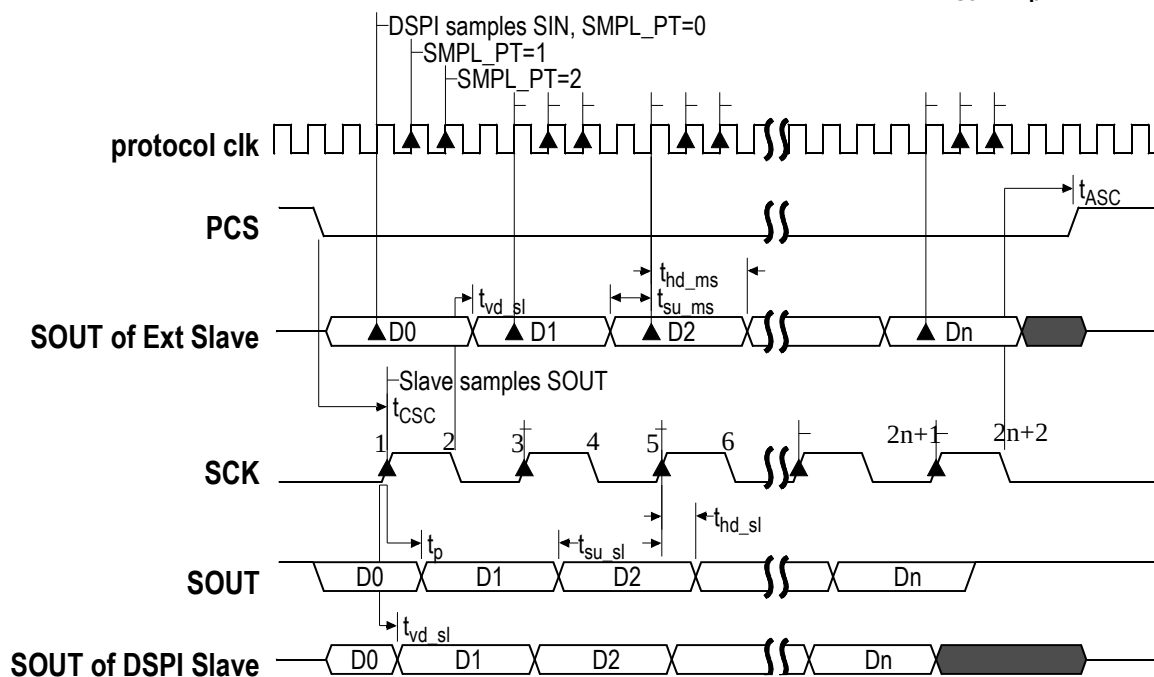
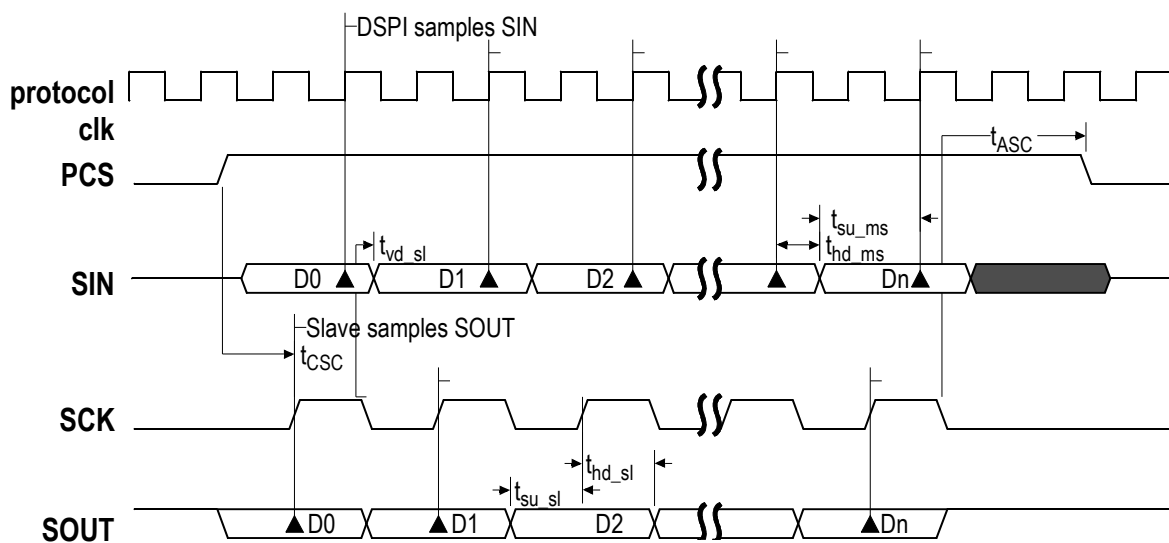
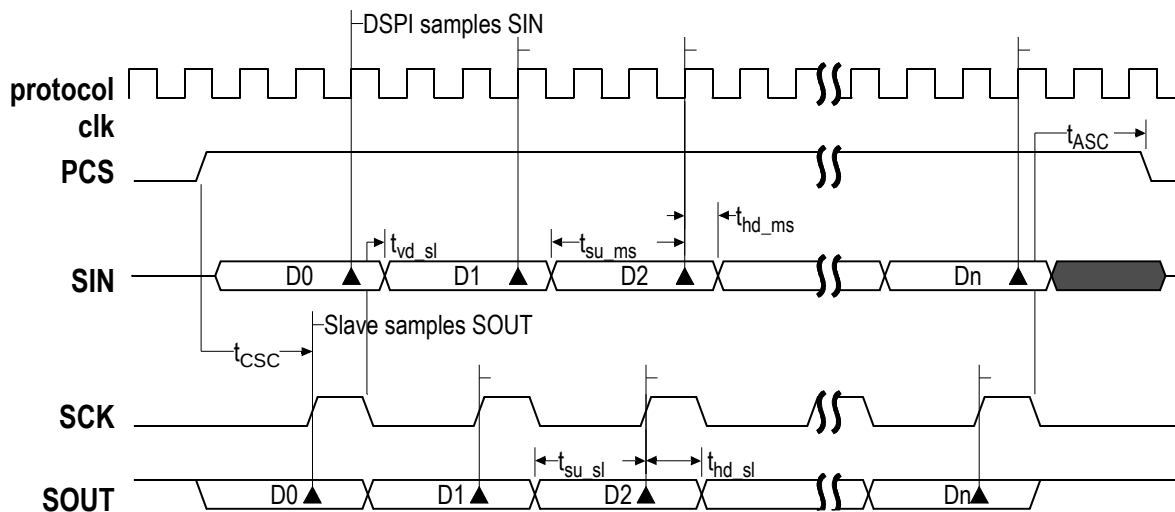
Figure 780. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$)Figure 781. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$)

Figure 782. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$)

48.4.6.4 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay must be greater or equal to half of the SCK period.**

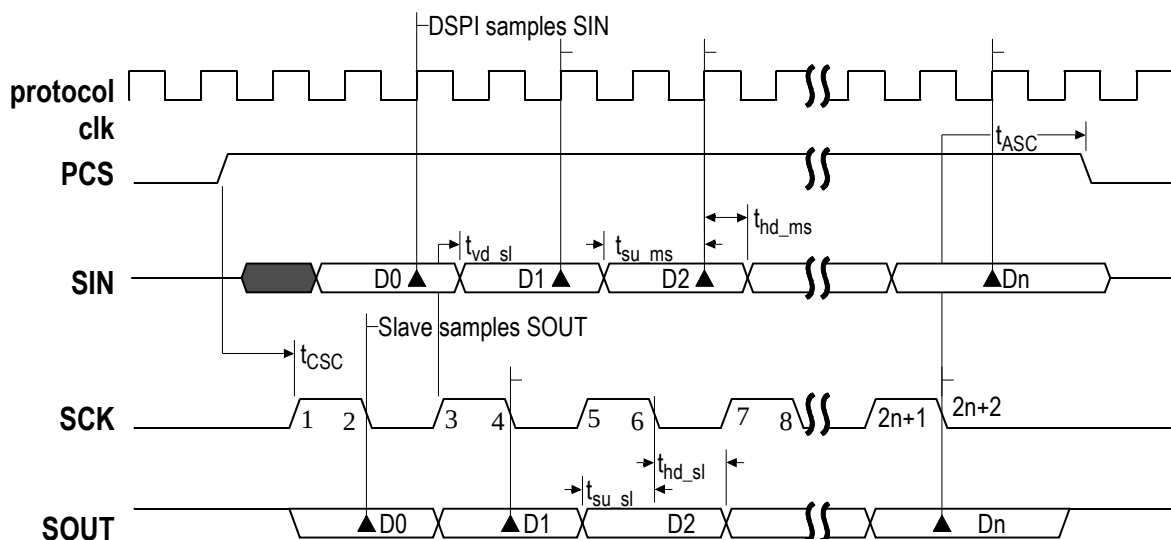
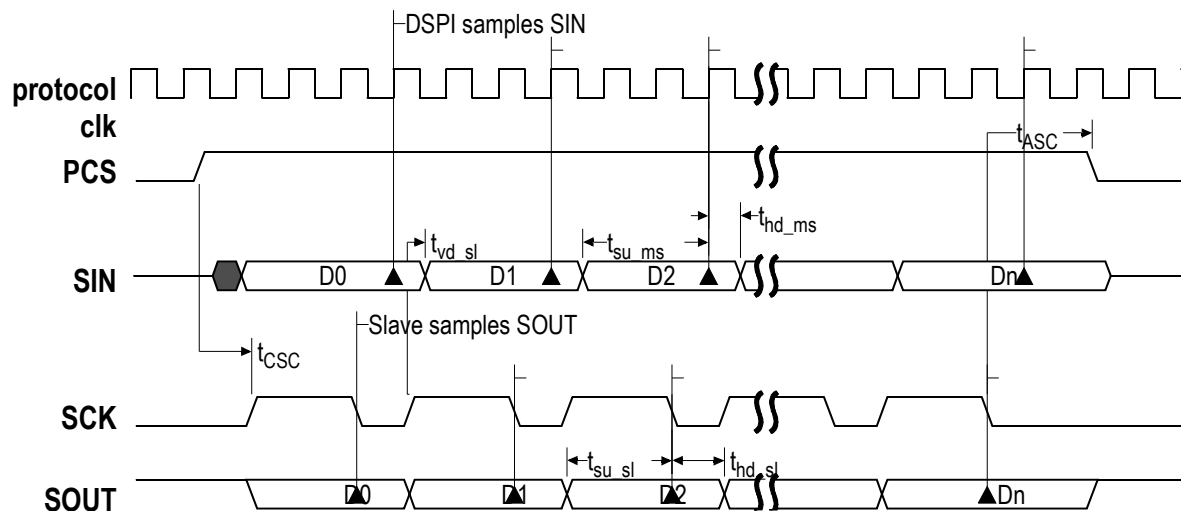
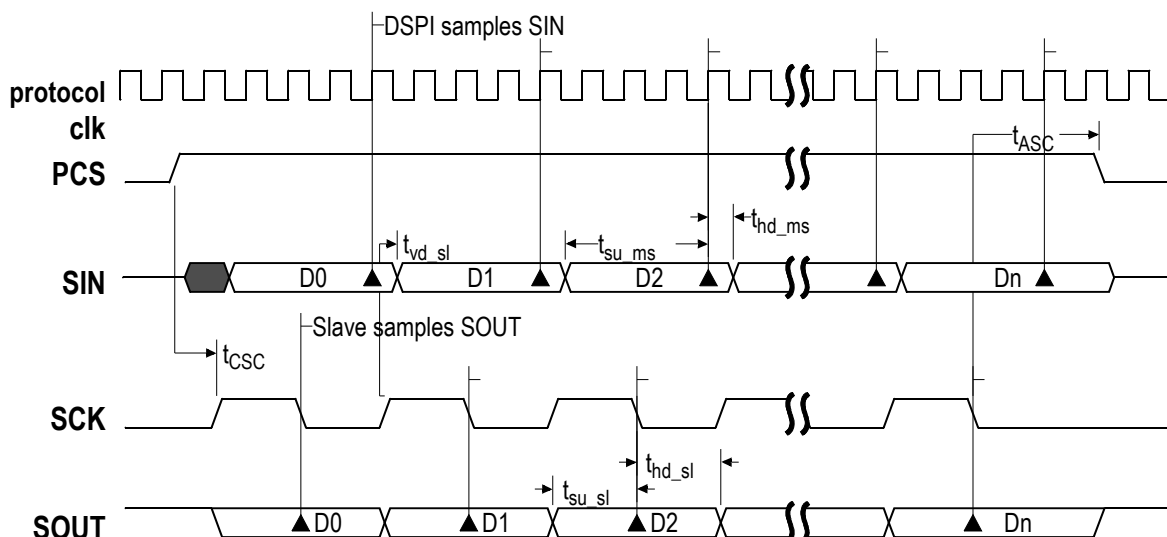
Figure 783. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$)

Figure 784. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$)Figure 785. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$)

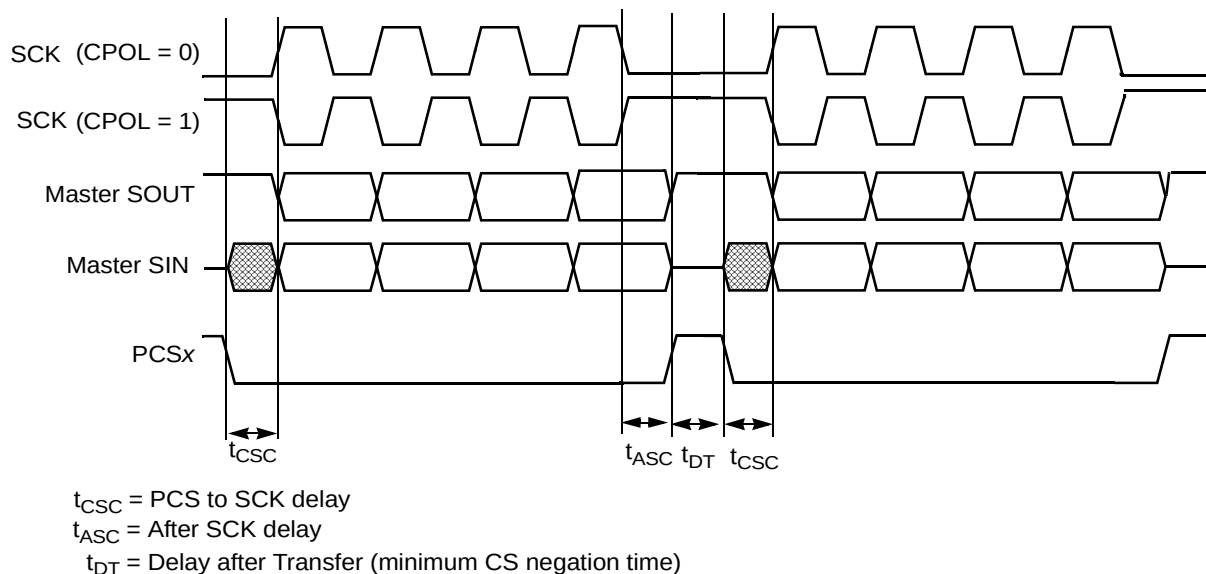
48.4.6.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both the following cases. The Continuous Selection Format is enabled for the SPI Configuration by setting the CONT bit in the SPI command. Continuous Selection is enabled for the DSI Configuration by setting the DCONT bit in the DSICR0.

The behavior of the PCS signals in the two configurations is identical so only SPI Configuration will be described. When the CONT bit = 0, the DSPI drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The timing diagram in [Figure 786](#) is for

two four-bit transfers with $CPHA = 1$ and $CONT = 0$.

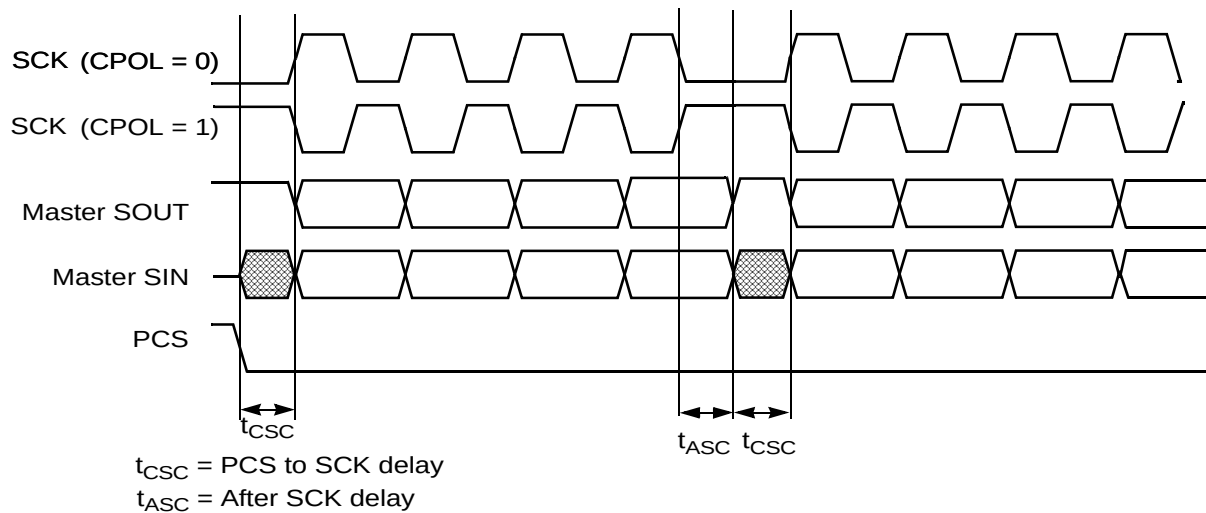
Figure 786. Example of non-continuous format ($CPHA = 1$, $CONT = 0$)



When the $CONT$ bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers.

[Figure 787](#) shows the timing diagram for two four-bit transfers with $CPHA = 1$ and $CONT = 1$.

Figure 787. Example of continuous transfer ($CPHA = 1$, $CONT = 1$)



When using DSPI with continuous selection follow these rules:

- all transmit commands must have the same PCSn bits programming.
- the CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.
- when transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHHR[CONT] bit de-asserted (in master mode) and the user software must provide sufficient frames in the TX_FIFO to be sent out (in slave mode) and the master de-asserts the PCSn at end of transmission of last frame.
- The PUSHHR[CONT]/DSICR0[DCONT] bits must be de-asserted before asserting MCR[HALT] bit (in master mode). This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] bit during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from RUNNING to STOPPED state.

Note: You must fill the TXFIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TXFIFO becomes empty.

When operating in slave mode, ensure that when the last-entry in the TXFIFO is completely transmitted (that is the corresponding TCF flag is asserted and TXFIFO is empty), the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

48.4.6.6 Fast Continuous Selection Format

The Fast Continuous Selection Format functions similar to the Continuous Selection Format except that the inter command delays, t_{ASC} and t_{CSC} , can be masked out and are not inserted by the hardware.

Note: The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled. Masking of delays is not allowed in DSI and CSI configurations or if the transfer is non-continuous.

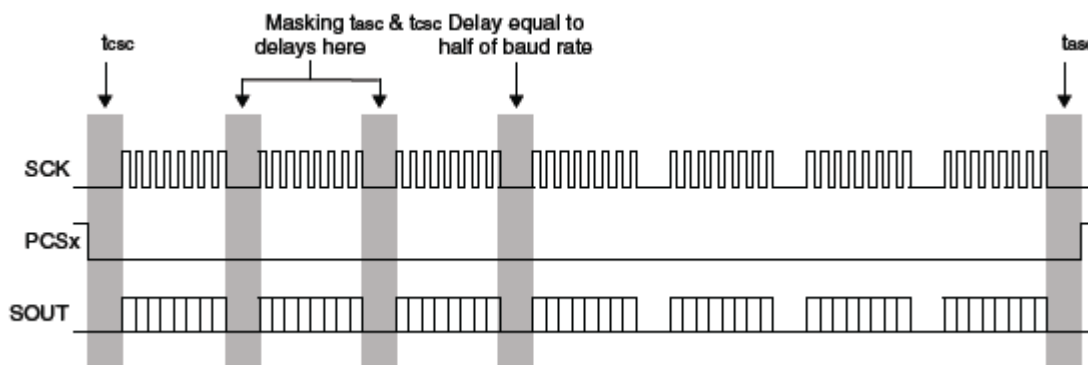
The Fast Continuous Selection Format is enabled by writing '1' into FCPCS bit of the MCR. When this bit is asserted, MASC and MCSC bits of the PUSHHR perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT bit asserted. In order to avoid these delays and speed up the transfer process, the software can simply mask these delays while programming the command in the PUSHHR.

While masking the delays, the software must follow the following masking rules, else correct operation is not guaranteed.

- MASC bit masks the “After SCK” delay for the current frame.
- MCSC bit masks the “PCS to SCK” delay for the next frame.
- “After SCK” (t_{ASC}) delay must not be masked when the current frame is the last frame in the continuous selection format.
- The “PCS to SCK” delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here, too.

Figure 788 shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

Figure 788. Example of Fast Continuous Selection Format



In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

48.4.7 Continuous Serial Communications Clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous Serial Communications Clock regardless of the MCR[HALT] bit status. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the DSPI is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- When the DSPI is in DSI configuration, the CTAR specified by the DSICTAS field is used at all times.
- When the DSPI is in CSI configuration, the CTAR selected by the DSICTAS field is used initially.
 - At the start of a SPI frame transfer, the CTAR specified by the CTAS value for the frame is used.
 - At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

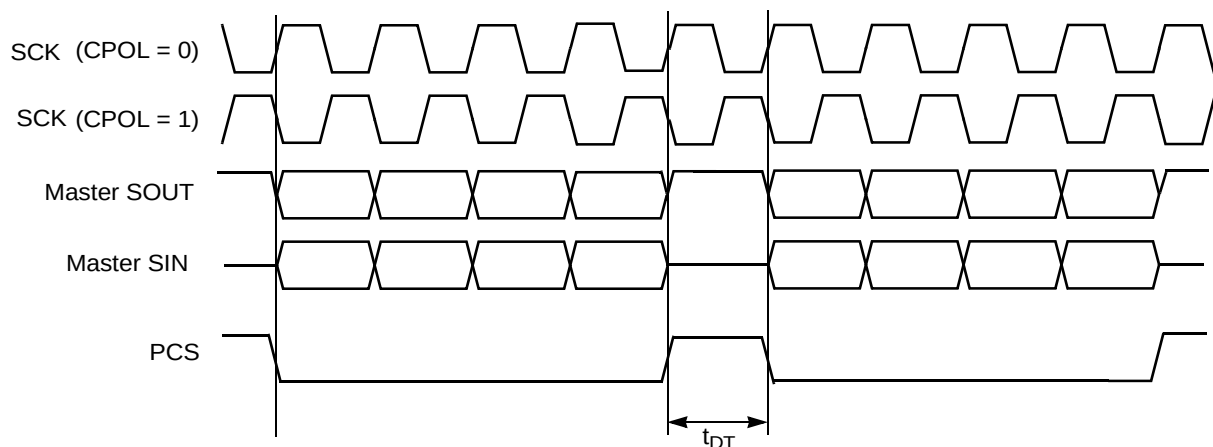
It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle.

When TSB configuration is enabled the t_{DT} is programmable from 1 to 64 SCK cycles. [Figure 789](#) is the timing diagram for Continuous SCK format with Continuous Selection disabled.

Note: When in Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TXFIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

Figure 789. Continuous SCK timing diagram (CONT = 0)



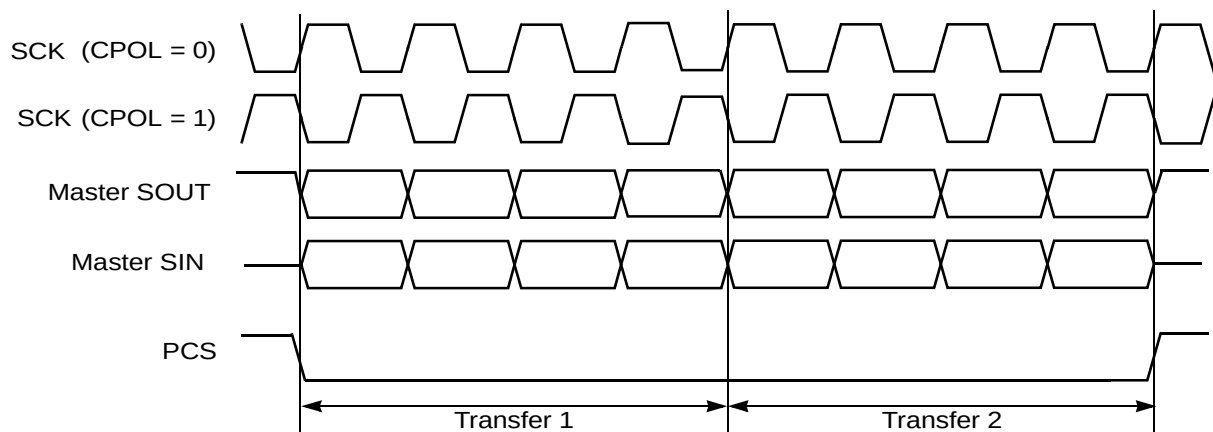
If the CONT bit in the TX FIFO entry is set or the DCONT in the DSICR0 is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with

PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO.
- Continuous SCK with CONT bit set and entering STOPPED state (refer to [Section 48.4.1, Start and Stop of DSPI transfers](#)).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

[Figure 790](#) shows the timing diagram for Continuous SCK format with Continuous Selection enabled.

Figure 790. Continuous SCK Timing Diagram (CONT = 1)



48.4.8 Slave mode operation constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the DSPI is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the SS signal is asserted and any time when transmit data is ready and SS signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR1/0 register.

If the SS negates before that last SCK edge, the data from shift register is lost.

48.4.9 Timed Serial Bus (TSB)

The DSPI can be programmed in Timed Serial Bus configuration by setting DSICR0[TSBC] bit. See DSPI DSI Configuration Register 0 (DSPI_DSICR0) for details.

TSB configuration provides the Micro Second Channel (MSC) downstream channel support.

The MSC upstream channel is not supported by the DSPI, but can be supported by any available Serial Communication Controller (SCI or UART) in the device.

To work in TSB mode the DSPI must be in master mode and in DSI (DCONF = 0b01) or CSI (DCONF = 0b10) configuration. Both Continuous and Non Continuous Serial Communication Clock (controlled by the DSPI_MCR[CONT_SCKE] bit) are supported in the TSB mode.

Figure 791 shows the signals used in the TSB interface.

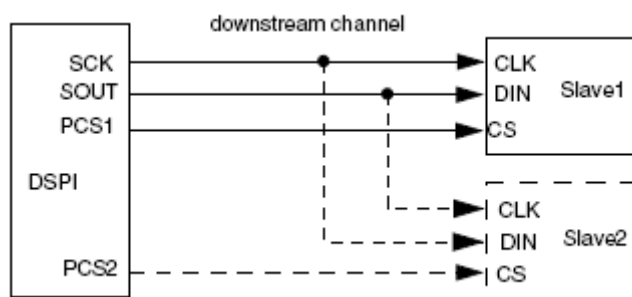
In the TSB configuration the DSPI is able to send from 4 to 66 bits MSC data frames (4 to 64 serialized data bits and up to 2 Data Selection zero bits). The serialized data bits source can be either:

- the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR1/0), written by the host software,
- parallel input pin states latched into the DSPI DSI Serialization Data Register (DSPI_SDR1/0).

DSICR0[TXSS] bit or DSPI_SSR1/0 register bits define the source of the data.

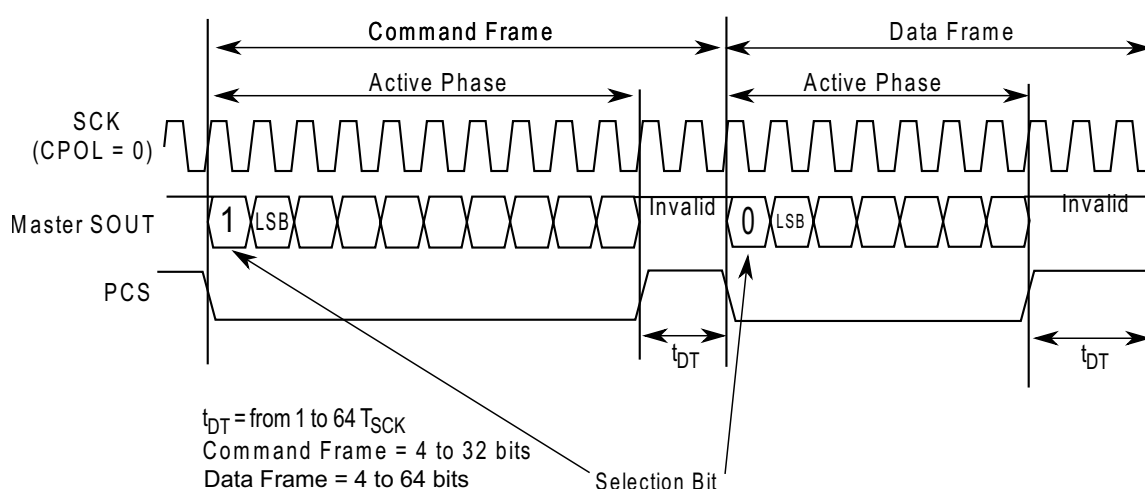
The Least Significant Bits of the DSPI_ASDR0 or DSPI_SDR0 registers are selected to be serialized if the data frame is set to less than 32 bits.

Figure 791. DSPI usage in the TSB Configuration



The PCS signals are driven together with SOUT. The t_{CSC} and t_{ASC} delays are not available. Delay after Transfer (DT) is set in SCK clock periods as a binary number formed by concatenation of the DSPI_CTARn PDT and DT fields plus one, allowing to set DT from 1 to 64 serial clock periods. DT field provides least significant bits and PDT field provides most significant bits of the Delay after Transfer.

Figure 792. TSB Downstream Frames



[Figure 792](#) shows the two types of MSC downstream frames: command frame, and data frame.

The first transmitted bit, called the selection bit, determines the frame type:

- The selection bit “0” indicates a data frame
- The selection bit “1” indicates a command frame

Data frame may contain up to 2 selection bits to support two external slave devices, (so called dual receiver configuration) or no selection bits at all.

The command frame can be written by software, through SPI TX FIFO, using one or two FIFO entries with help of the CONT bit or MCR[XSPI]. The data frame consists of up to 64 bits from the SDR1/0 or ASDR1/0 registers and up to two zero selection bits. Number of data bits in the data frame is defined by the DSICR1[TSBCNT] field.

The selection bit of the MSC command frames (1) can be implemented by software.

The selection bits in the data frames are enabled by DSPI_DSICR1 DSE0 and DSE1 bits. Each DSEn bit set increases the data frame size by one bit.

To comply with MSC specification, set DSPI_CTARn[LSBFE] to transmit least significant bit first.

Regardless of the LSBFE bit setting, the Data Frame Selection Bits, if enabled, are always transmitted first, before the corresponding data subframes.

48.4.9.1 MSC Dual Receiver Support with PCS Switchover

When in TSB mode it is possible to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame. The bit, at which this switchover occurs, is defined by FMSZ field of the DSPI_CTARn register, selected by DSICTAS field of the DSICR0 register. If DSICR1[DSI64E] is enabled, then the bit at which the PCS switch occurs, is defined by the concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields.

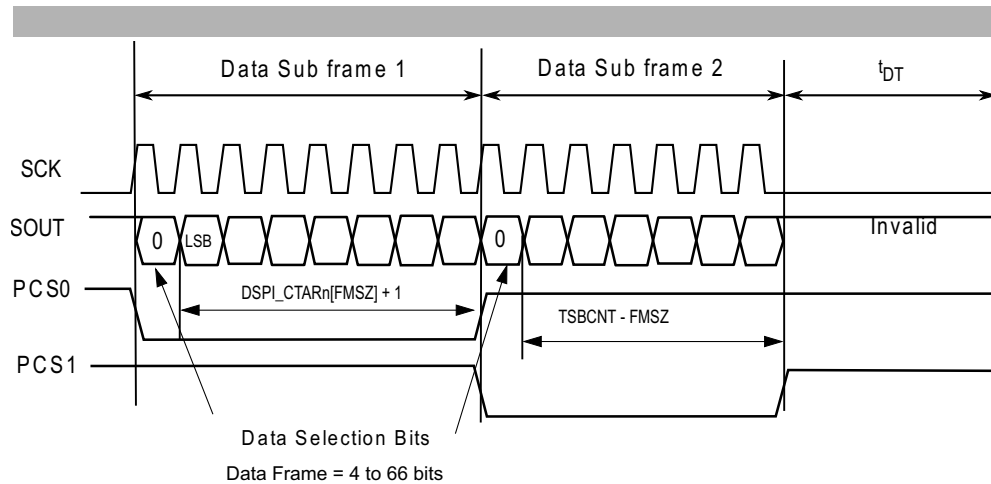
Number of the bits, not including the Data Selection Bit, in the first part of the frame is equal to value of the FMSZ field plus one. (This value changes to {FMSZ4,FMSZ} plus one, when DSICR1[DSI64E] is enabled). During this part of the frame the PCS signal levels are controlled by DSPI_DSICR0 DPCS_n bits, after that by DSPI_DSICR1 DPCS_{1_n} bits.

The PCS switchover occurs at the driving edge of the SCK clock output.

The second Data Selection Bit is inserted after the PCS switchover if enabled.

The Data Frame with PCS switchover is shown in [Figure 793](#).

Figure 793. TSB Data Frame Format for MSC Dual Receiver Operation



48.4.10 Interleaved TSB (ITSB) Mode

This mode is similar to the TSB in all aspects except for the differences mentioned in [Table 826](#). In the Interleaved TSB or ITSB mode, frame transmission is governed by the following rules:

- Transmission of frames is done on a periodic trigger that can be generated internal to the DSPI. The trigger period is calculated as follows:
 - Trigger Period = 64 bits (maximum data frame size) + 2 selection bits (maximum bits in data frame) + t_{PT}
 - t_{PT} is the passive frame time or dead time inserted to have constant trigger period.
 - The user software shall configure the trigger period and trigger source.
- If the previously transmitted frame was a SPI frame or the TX FIFO (or CMD FIFO) is empty, the frame transmitted on subsequent trigger will always be a DSI frame.
- If the previously transmitted frame was not a SPI frame and the TX_FIFO (and CMD FIFO) is not empty, the frame transmitted on subsequent trigger will always be a SPI frame.

This mode does not change the way DSI and SPI frame transmissions are done except that ITSB mode interleaves the source (SPI or DSI) in such a way to prevent back to back SPI frames. All features of SPI and DSI mode can be used except for the Continuous Frame Selection format (PUSHR[CONT] and DSICR0[DCONT] bits cannot be asserted).

MCR[MTFE] should be set to 0 during ITSB Mode of operation, since MicroSecond Channel upstream operation is not supported.

[Table 826](#) lists the differences between the TSB and ITSB modes.

Table 826. Differences between TSB and ITSB modes

Mode attribute	TSB mode	ITSB mode
Frame priority	Priority available to frames from SPI. Back to back SPI frames are sent out.	No priority. Messages are interleaved such that there are no back-to-back SPI frames
Frame selection bit	Selection bit inserted for DSI frames only. Selection bit for SPI frames is inserted by software.	Selection bit for both DSI and SPI frames inserted automatically. Encoding of selection bit same as TSB mode (0 = DSI; 1 = SPI).
Frame transmission	Occasionally triggered transmission (that is trigger is received via hardware trigger input).	Periodically triggered transmission (that is frames are transmitted continuously on every trigger).
Frame gap	Gap between frames is configurable. Time between frames is t_{DT} which is configurable from 1 to 64 baud rate clock cycles.	Gap between frames is inserted automatically as the whole operation is based on a periodic trigger. If a frame has less than 64-bits to be transmitted, dead or passive time is automatically inserted by waiting for the next trigger pulse. Time t_{DT} is not used in this mode as it can be made part of the trigger period (as passive time).

However, in both TSB and ITSB mode, separate frame completion interrupts will be available to indicate frame transmission completion from SPI and DSI. MSC dual receiver support with PCS switchover is also supported in ITSB mode. See [Section 48.4.9.1, MSC Dual Receiver Support with PCS Switchover](#) for more details.

The ITSB mode is suitable to implement the Downstream Channel for the MSC Bus because of the ITSB mode features listed in the table above.

48.4.10.1 Configuring DSPI for ITSB mode

The following steps give the recommended way to initialize the DSPI for the ITSB mode of operation:

- The DSPI should be put in HALT mode (by asserting the MCR[HALT] bit) before programming the registers.
- Set the DSICR0[TSBC] and DSICR0[ITSB] bits. The DSICR0[TRRE] and DSICR0[CID] bits should be cleared. DSICR0[DCONT] should not be used in ITSB mode. If DSICR0[ITSB] bit is set without setting DSICR0[TSBC], then the DSPI operates in the Normal Mode depending on MCR[DCONF] bit.
- Depending on the source of trigger, set the values in the DSICR0[TRG] bit.
- Enable frame completion interrupts for either any frame completion (TCF) or separate frame completion (DSITCF and SPITCF) as required by application.
- When remainder configuration of DSPI is complete, clear the MCR[HALT] bit to allow DSPI to start operations.

Once configured and DSPI is brought out of HALT, the ITSB mode shall start sequencing the frames from either DSI or SPI as per the rules mentioned in [Section 48.4.10, Interleaved TSB \(ITSB\) Mode](#).

48.4.10.2 Using ITSB mode for MSC downstream channel

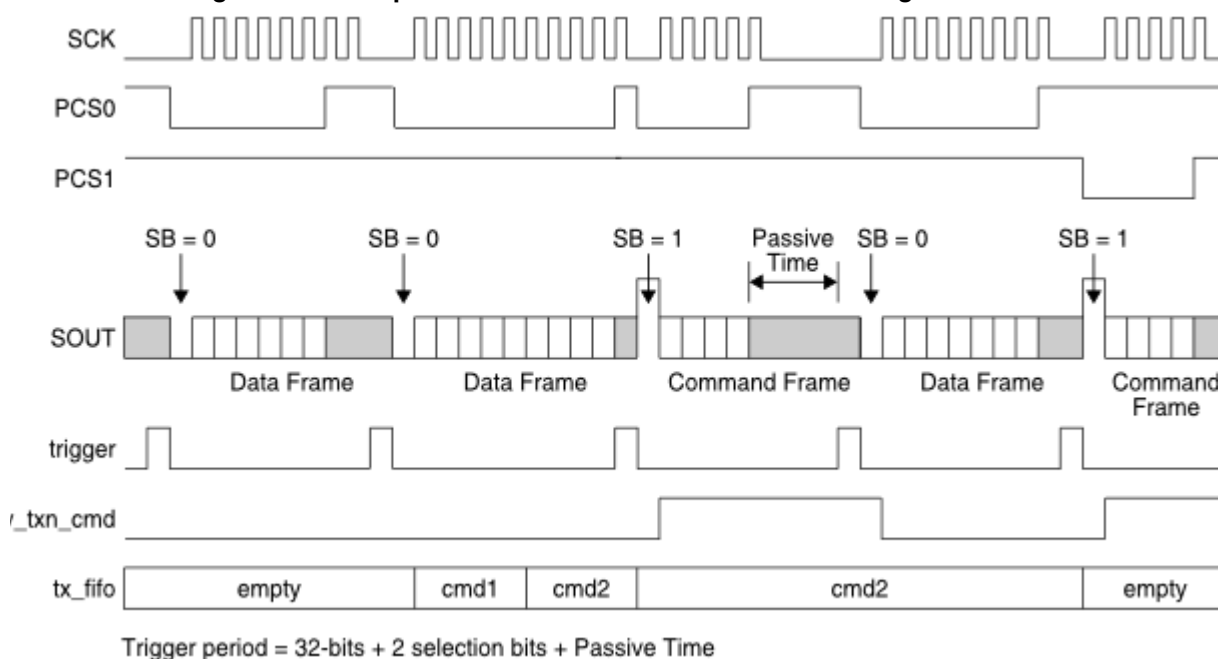
MSC Downstream Channel uses two types of frames: command frames and data frames. The Downstream operation of these frames require:

- Data frames can be sent back to back.
- Back to back command frames are not a legal scenario.
- If command frame has less bits than data frame, a gap is inserted to ensure next data frame starts at known reference time.
- In dual MSC ASIC, some bits of the data frame (which can be up to 64 bits) goes to first ASIC and remainder bits to second ASIC.
- In dual MSC ASIC, if a command frame is sent to one ASIC, it must not change the timing of the next data frame.

The ITSB mode can help implement the MSC Downstream Channel by allowing the command frames to be sent via the SPI interface and data frames to be sent via the DSI interface. Once the DSPI has been configured to work in ITSB mode (See [Section 48.4.10.1, Configuring DSPI for ITSB mode](#)), the user software can program the DSI and SPI sides to send data and command frames respectively. The programming of frames for transmission via DSI or SPI is same as explained in [Section 48.4.4, Combined Serial Interface \(CSI\) configuration](#).

[Figure 794](#) shows a sample MSC Downstream transmission using the ITSB mode.

Figure 794. Sample MSC downstream transmission using ITSB mode



Brief description of operation includes:

- At the first trigger, since there are no command frames, a data frame is transmitted: to send a command frame at the start of operation, the user software must program a command frame in the TX_FIFO before enabling DSPI (MCR[HALT] = 0).
- The second trigger sets off the transmission of another data frame.
- Between second and third trigger, two command frames are pushed into the TX_FIFO.
- At third trigger, a command frame is transmitted as the TX_FIFO is not empty: since only 16 bits or less are transmitted, no transmission happens till next trigger. This is marked as the "Passive Time". The passive time is added in all frames after frame transmission completes and until next trigger pulse is detected.
- At fourth trigger, since the previous transmission was a command frame, a data frame is transmitted.
- At fifth trigger, since one command frame is pending in TX_FIFO and last transmitted frame was not a command frame, a command frame is transmitted.
- At sixth trigger (not shown), the process continues.

48.4.11 Parity generation and check

The DSPI module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/control is done on frame basis. The register fields setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The DSPI module can be programmed to stop SPI or/and DSI frame transmission in case of a frame reception with parity error.

48.4.11.1 Parity for SPI frames

When the DSPI is in the master mode the parity generation is controlled by PE and PP bits of the TX FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the DSPI stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode, the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

48.4.11.2 Parity for DSI frames

When DSPI is in Master Mode, parity generation is controlled by PE and PP bits of the DSPI_DSICR0 register similar to the SPI frames. The parity is calculated and checked for each DSI frame. (DSPI_DSICR0[DCONT] bit has no effect on parity generation. In slave mode, the parity is controlled by the PE and PP bits of the CTAR1 register.

If the parity error occurs for received DSI frame, the DSPI_SR[DPEF] bit is set. If DSPI_DSICR0[PES] bit is set, the DSPI stops DSI frames transmission. To resume DSI operation, clear the DSPI_SR[DPEF] or the DSPI_DSICR0[PES] bits.

48.4.12 Interrupts/DMA requests

Certain DSPI conditions can generate only interrupt requests and other DSPI conditions can generate either interrupt requests or DMA requests. [Table 827](#) lists these conditions.

Table 827. Interrupt and DMA request conditions

DSPI condition	Flag	Request type	
		Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	—
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	—
Transfer Complete	TCF	Yes	—
CMD Transfer Complete	CMDTCF	Yes	—
SPI Transfer Complete	SPITCF	Yes	—
DSI Transfer Complete	DSITCF	Yes	—
TX FIFO Underflow	TFUF	Yes	—
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	—
SPI Parity Error	SPEF	Yes	—
DSI Parity Error	DPEF	Yes	—
DSI Deserialized Data Match	DDIF	Yes	Yes

Each condition has a flag bit in the DSPI Status Register (SR) and a Request Enable bit in the DSPI DMA/Interrupt Request Select and Enable Register (RSER). The TX FIFO Fill Flag (TFFF) and RX FIFO Drain Flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the RSER.

The DSPI module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

48.4.12.1 End of Queue interrupt request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI command is set and the EOQF_RE bit in the RSER is set.

When MCR[XSPI] is enabled and the EOQ bit in the executing SPI command is set, the End of Queue Request will only be generated once the last Data frame in the Command Cycle has been transmitted.

Note: This interrupt request is generated when the last bit of the SPI frame with EOQ bit set is transmitted.

48.4.12.2 Transmit FIFO Fill interrupt or DMA request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

Note: TFFF flag clears automatically when DMA is used to fill TXFIFO.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TXFIFO:

- Wait until TFFF = 1
- Write data to PUSHR using CPU.
- Clear TFFF by writing a '1' to its location. If TX FIFO is not full, this flag will not clear.

48.4.12.3 Command FIFO Fill interrupt or DMA request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled and hence TX FIFO and CMD FIFO can be filled independently. If MCR[XSPI] is disabled, then 'TX FIFO Fill Interrupt or DMA Request' will suffice to fill both FIFO's since both FIFO's must be filled simultaneously.

Note: CMDFFF flag clears automatically when DMA is used to fill CMD FIFO.

To clear CMDFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill CMD FIFO:

- Wait until CMDFFF = 1
- Write data to Command field of PUSHR using CPU.
- Clear CMDFFF by writing a '1' to its location. If CMD FIFO is not full, this flag will not clear.

48.4.12.4 Transmit FIFO Invalid Write interrupt request

The Transfer Fifo Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transfer Fifo Invalid Write Request is generated for the above condition when TFIWF_RE bit is set in the RSER.

48.4.12.5 Transfer Complete interrupt request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

48.4.12.6 Command Transfer Complete interrupt request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when the CMDTCF_RE bit is set in the RSER.

48.4.12.7 SPI Transfer Complete interrupt request

The SPI Transfer Complete Request indicates the end of transfer of an SPI frame in any Transmission Mode which uses DSPI in CSI Mode. The SPI Transfer Complete Request is generated when SR[SPITCF] flag asserts and SPITCF_RE bit is set in the RSER.

48.4.12.8 DSI Transfer Complete interrupt request

The DSI Transfer Complete Request indicates the end of transfer of a DSI frame in any Transmission Mode which uses DSPI in CSI Mode. The DSI Transfer Complete Request is generated when SR[DSITCF] flag asserts and DSITCF_RE bit is set in the RSER.

48.4.12.9 Transmit FIFO Underflow interrupt request

The Transmit FIFO Underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the DSPI, operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

48.4.12.10 Receive FIFO Drain interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

48.4.12.11 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

48.4.12.12 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

48.4.12.13 DSI Frame Parity Error Interrupt Request

The DSI Frame Parity Error Flag indicates that a DSI frame with parity error has been received. The DPEF_RE bit in the DSPI_RSER must be set for the interrupt request to be generated.

48.4.12.14 Deserialized Data Match Interrupt or DMA Request

The Deserialized Data Match Flag (DDIF) indicates that a DSI frame matches DSPI_DPIR1/0 data, masked with DSPI_DIMR1/0, has been received. This Request is generated if DDIF_RE bit in the DSPI_RSER is set. The DDIF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

48.4.13 Power saving features

The DSPI supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory-mapped logic

48.4.13.1 Stop mode (External Stop Mode)

The DSPI supports the stop mode protocol. When a request is made to enter external stop mode, the DSPI block acknowledges the request. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it is ready to have its clocks shut off. While the clocks are shut off, the DSPI memory-mapped logic is not accessible. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

48.4.13.2 Module disable mode

Module disable mode is a block-specific mode that the DSPI can enter to save power. The host CPU can initiate the module disable mode by setting the MDIS bit in the MCR. The module disable mode can also be initiated by hardware. A power management block can initiate the module disable mode by asserting the DOZE mode signal while the DOZE bit in the MCR is set.

When the MDIS bit is set or the DOZE mode signal is asserted while the DOZE bit is set, the DSPI negates Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, DSPI is said to have entered Module Disable Mode. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. If implemented, the Clock Enable signal can stop the clock to the non-memory-mapped logic. When Clock Enable is negated, the DSPI is in a dormant state, but the memory-mapped registers are still accessible. Certain read or write operations have a different effect when the DSPI is in the module disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Likewise, writing to the FIFO Push Register does not change the state of the TX FIFO or CMD FIFO. Clearing of the FIFOs has no effect in the module disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the module disable mode. In the module disable mode, all status bits and register flags in the DSPI return the correct values when read, but writing to them has no effect. Writing to the TCR during module disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the module disable mode.

48.5 Initialization/application information

This section describes how to initialize the DSPI module.

48.5.1 Managing DSPI queues

The queues are not part of the DSPI, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI Configuration.

1. When DSPI executes last command word from a queue, the EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is cleared to indicate the STOPPED state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a '1' to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

48.5.2 Switching master and slave mode

When changing modes in the DSPI, follow the steps below to guarantee proper operation.

1. Halt the DSPI by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a '1' to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable the DSPI by clearing MCR[HALT].

48.5.3 Initializing DSPI in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the DSPI is enabled by clearing MCR[HALT]. It should be ensured that DSPI Slave is enabled before enabling DSPI Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

48.5.4 Baud rate settings

[Table 828](#) shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is clear.

Note: The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the clocking chapter for the frequency used to drive this module in the device.

Table 828. Baud rate values (bps)

		Baud rate divider prescaler values (DSPI _n _CTAR[PBR])			
		2	3	5	7
Baud Rate scaler values (DSPI _n _CTAR[BR])	2	25.0 MHz	16.7 MHz	10.0 MHz	7.14 MHz
	4	12.5 MHz	8.33 MHz	5.00 MHz	3.57 MHz
	6	8.33 MHz	5.56 MHz	3.33 MHz	2.38 MHz
	8	6.25 MHz	4.17 MHz	2.50 MHz	1.79 MHz
	16	3.12 MHz	2.08 MHz	1.25 MHz	893 kHz
	32	1.56 MHz	1.04 MHz	625 kHz	446 kHz
	64	781 kHz	521 kHz	312 kHz	223 kHz
	128	391 kHz	260 kHz	156 kHz	112 kHz
	256	195 kHz	130 kHz	78.1 kHz	55.8 kHz
	512	97.7 kHz	65.1 kHz	39.1 kHz	27.9 kHz
	1024	48.8 kHz	32.6 kHz	19.5 kHz	14.0 kHz
	2048	24.4 kHz	16.3 kHz	9.77 kHz	6.98 kHz
	4096	12.2 kHz	8.14 kHz	4.88 kHz	3.49 kHz
	8192	6.10 kHz	4.07 kHz	2.44 kHz	1.74 kHz
	16384	3.05 kHz	2.04 kHz	1.22 kHz	872 Hz
	32768	1.53 kHz	1.02 kHz	610 Hz	436 Hz

48.5.5 Delay settings

[Table 829](#) shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (t_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

Note: The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the clocking chapter for the frequency used to drive this module in the device.

This table does not apply for TSB Continuous mode.

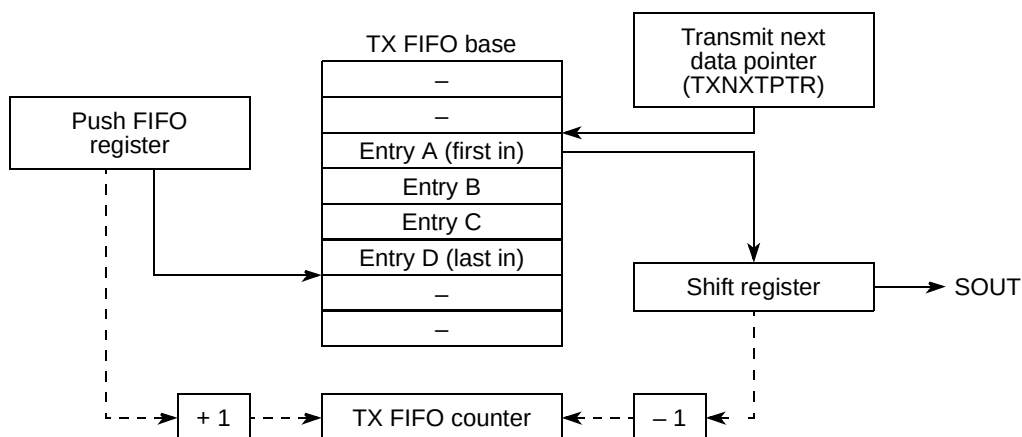
Table 829. Delay values

		Delay prescaler values (DSPIn_CTAR[PBR])			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 μ s
	32	320.0 ns	960.0 ns	1.6 μ s	2.2 μ s
	64	640.0 ns	1.9 μ s	3.2 μ s	4.5 μ s
	128	1.3 μ s	3.8 μ s	6.4 μ s	9.0 μ s
	256	2.6 μ s	7.7 μ s	12.8 μ s	17.9 μ s
	512	5.1 μ s	15.4 μ s	25.6 μ s	35.8 μ s
	1024	10.2 μ s	30.7 μ s	51.2 μ s	71.7 μ s
	2048	20.5 μ s	61.4 μ s	102.4 μ s	143.4 μ s
	4096	41.0 μ s	122.9 μ s	204.8 μ s	286.7 μ s
	8192	81.9 μ s	245.8 μ s	409.6 μ s	573.4 μ s
	16384	163.8 μ s	491.5 μ s	819.2 μ s	1.1 ms
	32768	327.7 μ s	983.0 μ s	1.6 ms	2.3 ms
	65536	655.4 μ s	2.0 ms	3.3 ms	4.6 ms

48.5.6 Calculation of FIFO pointer addresses

Complete visibility of the TX and CMD and RX FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and a memory-mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory-mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). [Figure 795](#) illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over to the CMD FIFO and RX FIFO. See [Section 48.4.2.4, Transmit First In First Out \(TX FIFO\) buffering mechanism](#), [Section 48.4.2.5, Command First In First Out \(CMD FIFO\) buffering mechanism](#) and [Section 48.4.2.6, Receive First In First Out \(RX FIFO\) Buffering Mechanism](#) for details on the FIFO operation.

Figure 795. TX FIFO pointers and counter



48.5.6.1 Address calculation for the first-in entry and last-in entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{Equation 31} \quad \text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXPTR})$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Equation 32} \quad \text{Last-in EntryAddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXPTR} - 1) \bmod (\text{TXFIFOdepth})$$

TX FIFO Base – Base address of TX FIFO

TXCTR – TX FIFO Counter

TXNXPTR – Transmit Next Pointer

TX FIFO Depth – Transmit FIFO depth, implementation specific

48.5.6.2 Address calculation for the first-in entry and last-in entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

$$\text{Equation 33} \quad \text{First-in EntryAddress} = \text{CMDFIFOBase} + (4 \times \text{CMDNXPTR})$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

Equation 34 Last-in

$$\text{EntryAddress} = \text{CMDFIFOBase} + 4 \times (\text{CMDCTR} + \text{CMDNXTPTR} - 1) \bmod(\text{CMDFIFOdepth})$$

CMD FIFO Base – Base address of CMD FIFO

CMDCTR – CMD FIFO Counter

CMDNXTPTR – Command Next Pointer

CMD FIFO Depth – Command FIFO depth, implementation specific

48.5.6.3 Address calculation for the first-in entry and last-in entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

Equation 35 First-in
$$\text{EntryAddress} = \text{RX FIFOBase} + (4 \times \text{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

Equation 36 Last-in
$$\text{EntryAddress} = \text{RX FIFOBase} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1) \bmod(\text{RXFIFOdepth})$$

RX FIFO Base – Base address of RX FIFO

RXCTR – RX FIFO counter

POPNXTPTR – Pop Next Pointer

RX FIFO Depth – Receive FIFO depth, implementation specific

49 LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

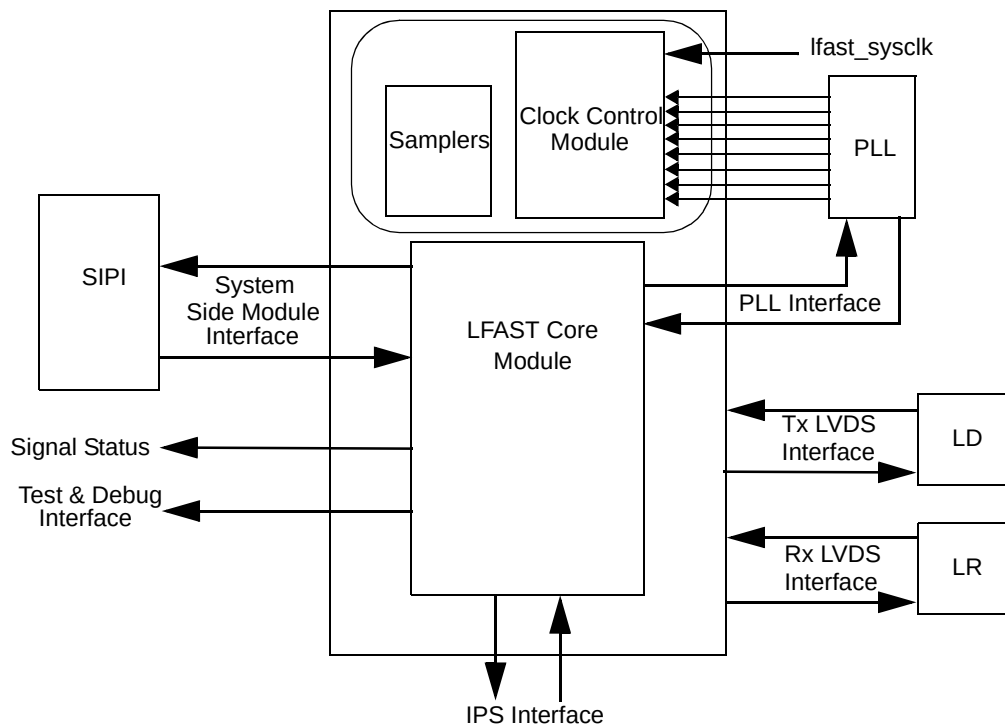
49.1 Introduction

This chapter describes the specifications of the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in dual mode (software configurable master/slave operation) for interprocessor communications.

49.2 Block diagram

Figure 796 depicts LFAST interaction with other modules on the device.

Figure 796. LFAST block diagram



49.3 External signals

LFAST is a five pin interface with the following signals:

- lfast_sysclk
 - Reference clock of the LFAST master and slave
- txdatap/txdatan
 - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
 - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface with maximum data rate of 320 Mbps.

49.3.1 LFAST operating data rates

[Table 830](#) shows the operating modes for the LFAST interface.

Table 830. Operating modes for LFAST interface

Interface	Tx/Rx Data Rate
LFAST Transmit	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps
LFAST Receive	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps ($\text{lfast_sysclk} \div 4$ or $\text{lfast_sysclk} \div 2$) is referred to as low and data rates 312 Mbps and 320 Mbps is referred to as high data rates.

49.4 LFAST frame structure

LFAST frames are transmitted and received across a serial interface using a packeted asynchronous communication format.

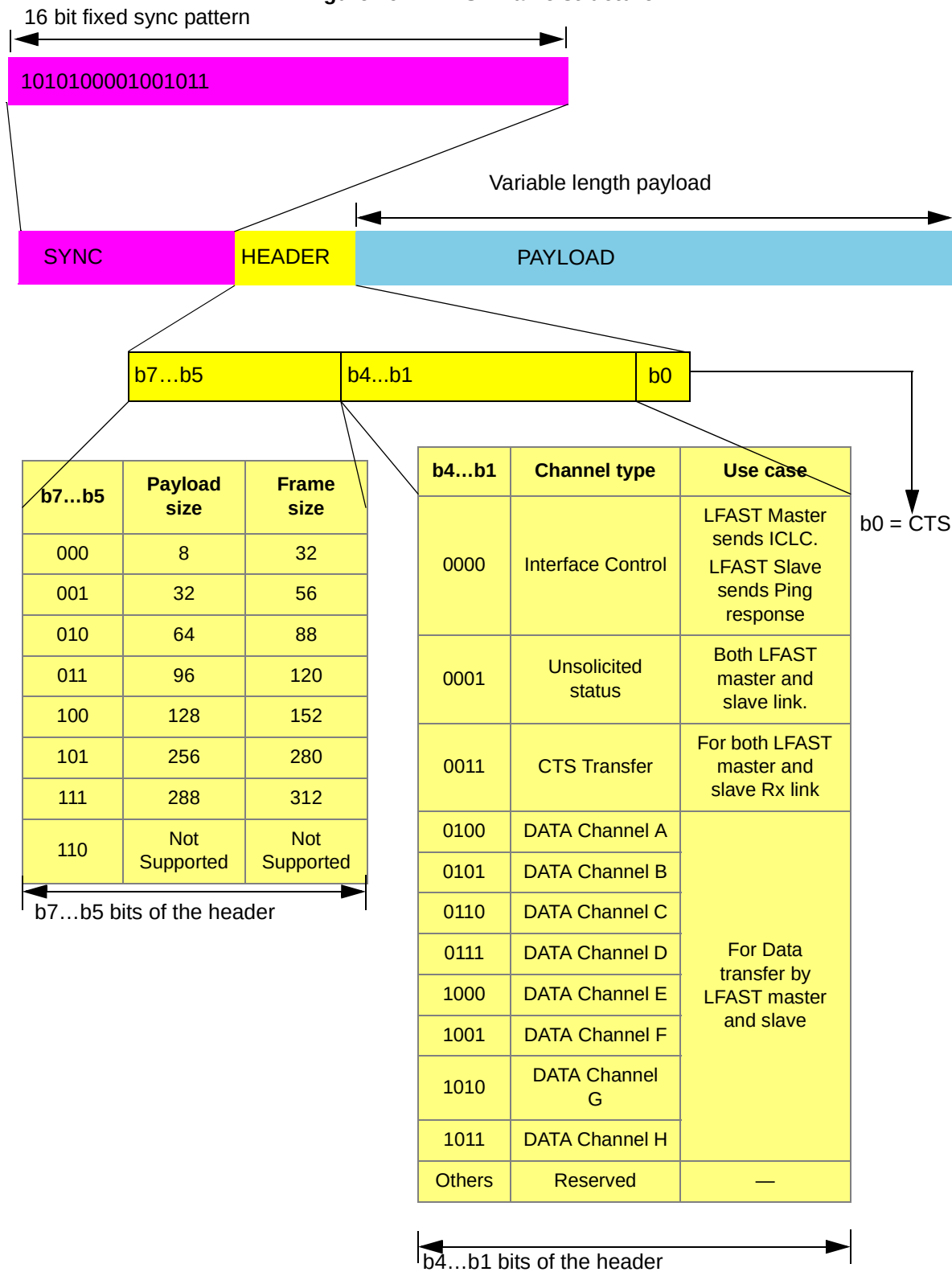
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length. A sync pattern is used to synchronize incoming data stream in LFAST module.

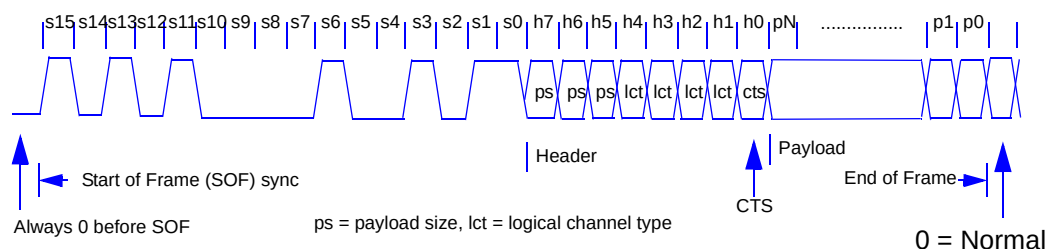
Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See [Figure 797](#) for details of the LFAST frame.

Figure 797. LFAST frame structure



The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

Figure 798. Serial frame structure



The frame structure is shown in [Figure 798](#) and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size according to [Table 831](#).

Table 831. Header payload sizes

b7-b5 (bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	—	—
111	288	312

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type and clear to send (CTS) frame.
- **Header - (b0):** CTS on the both LFAST master and slave devices.
- **Payload: Content dependent upon frame type.**
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode).

49.5 Features

- Supports dual mode (register configurable Master/Slave).
- Supports asynchronous data transfer rates up to a maximum of 320 Mbps
- Transmits and receives data, CTS, ICLC and unsolicited frames.
- Provision of five interrupts for Tx and Rx channels.
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports LFAST defined CTS controlled data transfer. No dependency between data transmission and reception unless CTS mode is enabled.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for data frame with variable payload sizes of 32, 64, 96, 128, 256 or 288 bits.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation in slave mode.
- Supports for detection of unsupported channel number and unsupported payload size.

49.6 Memory map and register definition

All registers are 32 bits wide.

49.6.1 Memory map

Table 832. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0000	LFAST Mode Configuration Register (MCR)	R/W	0000_0010	on page 1742
0004	LFAST Speed Control Register (SCR)	R/W	0001_0000	on page 1744
0008	LFAST Correlator Control Register (COCR)	R/W	0000_000E	on page 1745
000C	LFAST Test Mode Control Register (TMCR)	R/W	0000_0000	on page 1747
0010	LFAST Auto Loopback Control Register (ALCR)	R/W	0000_0000	on page 1748
0014	LFAST Rate Change Delay Control Register (RCDCR)	R/W	000F_0000	on page 1749
0018	LFAST Wakeup Delay Control Register (SLCR)	R/W	1201_5F02	on page 1750
001C	LFAST ICLC Control Register (ICR)	R/W	0000_0000	on page 1752
0020	LFAST Ping Control Register (PICR)	R/W	0000_80CA	on page 1753
0024–0028	Reserved	—	—	—
002C	LFAST Rx FIFO CTS Control Register (RFCR)	R/W	000F_0009	on page 1754

Table 832. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0030	LFAST Tx Interrupt Enable Register (TIER)	R/W	0000 0000	on page 1755
0034	LFAST Rx Interrupt Enable Register (RIER)	R/W	0000 0000	on page 1756
0038	LFAST Rx ICLC Interrupt Enable Register (RIIER)	R/W	0000 0000	on page 1757
003C	LFAST PLL Control Register (PLLCR)	R/W	0000_005C	on page 1759
0040	LFAST LVDS Control Register (LCR)	R/W	0000_5024	on page 1761
0044	LFAST Unsolicited Tx Control Register (UNSTCR)	R/W	0000_0000	on page 1763
0048	LFAST Unsolicited Tx Data Registers 8 (UNSTDR8)	R/W	0000_0000	on page 1764
004C	LFAST Unsolicited Tx Data Registers 7 (UNSTDR7)	R/W	0000_0000	on page 1764
0050	LFAST Unsolicited Tx Data Registers 6 (UNSTDR6)	R/W	0000_0000	on page 1764
0054	LFAST Unsolicited Tx Data Registers 5 (UNSTDR5)	R/W	0000_0000	on page 1764
0058	LFAST Unsolicited Tx Data Registers 4 (UNSTDR4)	R/W	0000_0000	on page 1764
005C	LFAST Unsolicited Tx Data Registers 3 (UNSTDR3)	R/W	0000_0000	on page 1764
0060	LFAST Unsolicited Tx Data Registers 2 (UNSTDR2)	R/W	0000_0000	on page 1764
0064	LFAST Unsolicited Tx Data Registers 1 (UNSTDR1)	R/W	0000_0000	on page 1764
0068	LFAST Unsolicited Tx Data Registers 0 (UNSTDR0)	R/W	0000 0000	on page 1764
006C–007C	Reserved	—	—	—
0080	LFAST Global Status Register (GSR)	R	0000_0000	on page 1764
0084	LFAST Ping Status Register (PISR)	R	0000_0000	on page 1766
0088–0090	Reserved	—	—	—
0094	LFAST Data Frame Status Register (DFSR)	R	0000_0000	on page 1766
0098	LFAST Tx Interrupt Status Register (TISR)	R/W	0000_0000	on page 1767
009C	LFAST Rx Interrupt Status Register (RISR)	R/W	0000_0000	on page 1768
00A0	LFAST Rx ICLC Interrupt Status Register (RIISR)	R/W	0000_0000	on page 1770
00A4	LFAST PLL and LVDS Status Register (PLLLSR)	R	0002_0003	on page 1771
00A8	LFAST Unsolicited Rx Status Register (UNSRSR)	R/W	0000_0000	on page 1772
00AC	LFAST Unsolicited Rx Data Register 8 (UNSRDR8)	R	0000_0000	on page 1773
00B0	LFAST Unsolicited Rx Data Register 7 (UNSRDR7)	R	0000_0000	on page 1773
00B4	LFAST Unsolicited Rx Data Register 6 (UNSRDR6)	R	0000_0000	on page 1773
00B8	LFAST Unsolicited Rx Data Register 5 (UNSRDR5)	R	0000_0000	on page 1773
00BC	LFAST Unsolicited Rx Data Register 4 (UNSRDR4)	R	0000_0000	on page 1773
00C0	LFAST Unsolicited Rx Data Register 3 (UNSRDR3)	R	0000_0000	on page 1773
00C4	LFAST Unsolicited Rx Data Register 2 (UNSRDR2)	R	0000_0000	on page 1773

Table 832. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
00C8	LFAST Unsolicited Rx Data Register 1 (UNSRDR1)	R	0000_0000	on page 1773
00CC	LFAST Unsolicited Rx Data Register 0 (UNSRDR0)	R	0000_0000	on page 1773

The conventions in [Figure 799](#) and [Table 833](#) serve as a key for the register summary and individual register diagrams.

Figure 799. Key to Register Fields

Always reads 1	1	Always reads 0	0	R/W bit	BIT	Read-only bit	BIT	Write-only bit	BIT	Write 1 to clear	BIT w1c	Self-clear bit	0 BIT
----------------	----------	----------------	----------	---------	------------	---------------	------------	----------------	------------	------------------	----------------	----------------	--------------

[Table 833](#) provides a key for register figures and tables and the register summary.

Table 833. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bits value (other than a hardware reset).
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but its cleared by the H/W once the operation defined by the bit is over
Reset Values	
0	Resets to zero.
1	Resets to one.
[signal_name]	Reset value is determined by polarity of indicated signal.

49.6.2 Register descriptions

49.6.2.1 Mode Configuration Register (MCR)

Figure 800. Mode Configuration Register (MCR)

Offset: 0000h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSEN ⁽²⁾	0	0	0	0	0	0	IPGDBG	0	0	0	0	0	0	0	LSSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0	0	0	0	0	0	0	0	TXARBD	CTSEN ⁽³⁾	0	DRFRST ⁽¹⁾	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Set by user software and cleared by system hardware.
2. Writable only once after the asynchronous reset.
3. Only writable when MCR[DRFEN] = 0.

Table 834. MCR field descriptions

Bits	Description
0 MSEN	LFAST Master or Slave mode Enable. This bit selects either the LFAST master or slave functionality. This bit can be written only once after the asynchronous reset. 0 Enable the modules LFAST Slave functionality only. 1 Enable the modules LFAST Master functionality only.
1-6	Reserved
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8-14	Reserved
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode. Refer to Section 49.10.2, Slow speed clock for more detailed description. 0 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.

Table 834. MCR field descriptions (continued)

Bits	Description
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS LR. 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface is not disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command frame. 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 Transmitter is Enabled.
19-26	Reserved
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 CTSEN	CTS Enable. This bit defines the Push-Pull mode of the LFAST devices receiver. This bit is used to enable/disable CTS mode of the Tx block. 0 CTS mode is disabled. Indicates that the device is in Push mode. The CTS bit of frames transmitted is 1. The CTS bit does not represent the status of Rx FIFO. 1 CTS mode is enabled. The CTS bit of all transmit frames is set when the Rx FIFO empty space is on or above higher threshold, and cleared when the Rx FIFO empty space is on or below lower threshold.
29	Reserved
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after reset. 0 No soft reset. 1 Soft reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

49.6.2.2 Speed Control Register (SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Figure 801. Speed Control Register (SCR)

Offset: 0004h								Access: User read/write									
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DRMD ⁽¹⁾
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	RDR	0	0	0	0	0	0	0	TDR	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

1. Only writable when MCR[DRFEN] = 0.

Table 835. SCR field descriptions

Bits	Description
0-14	Reserved
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. 0 S/W controls the Data Rate controller mode. In LFAST Slave the ICLC frames for rate change have no effect on the Data rate. 1 In LFAST Slave the reception of ICLC frame for rate change sets appropriate speed mode. In LFAST Master the SCR[DRMD] should be 0.
16 - 22	Reserved
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. For LFAST Master: – S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Tx interface. For LFAST Slave: – When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Rx interface. – The S/W can program this bit when SCR[DRMD] = 0. – This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is 312/320 Mbps.

Table 835. SCR field descriptions (continued)

Bits	Description
24-30	Reserved
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. For LFAST Master: – The S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Rx interface. For LFAST Slave: – When SCR[DRMD] = 1, the H/W writes this bit on receipt of an ICLC frame changing the speed mode of the slaves Tx interface. – The S/W can program this bit when SCR[DRMD] = 0. – This bit is cleared on MCR[DRFEN] = 0. 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is 312 /320 Mbps.

49.6.2.3 Correlator Control Register (COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Figure 802. Correlator Control Register (COCR)

Offset: 0008h				Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	SMPSEL ⁽¹⁾								0	0	0	0	0	0	0	0			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	0	0	0	0	0	0	0	0	0	0	0	0	CORRTH ⁽¹⁾				PHSSEL ⁽¹⁾		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0			

1. Only writable only when MCR[RXEN] = 0.

Table 836. COCR field descriptions

Field	Description
0-7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. 00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8-27	Reserved
28-30 CORRTH	Correlator threshold level. Defines the correlation threshold level. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. 08 phases 14 phases

49.6.2.4 Test Mode Control Register (TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes

Figure 803. Test Mode Control Register (TMCR)

Offset: 000Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	CLKTST	LPON	0	0	0	0	0	LPMOD ⁽¹⁾		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFRMTH ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Table 837. TMCR field descriptions

Bits	Description
0-5	Reserved
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command. 1Clock Test mode enabled 0Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1Loopback mode is enabled 0Loopback mode is disabled
8-12	Reserved

Table 837. TMCR field descriptions (continued)

Bits	Description
13-15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation Others - Reserved (should not be written by S/W)
16-31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

49.6.2.5 Auto Loopback Control Register (ALCR)

Figure 804. Auto Loopback Control Register (ALCR)

Offset: 0010h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LPCNTEN ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Table 838. ALCR field descriptions

Bits	Description
0-14	Reserved
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto predefined loopback frame transmission. 0 Infinite predefined loopback frame transmission enabled 1 Fixed count of predefined loopback frame transmission enabled
16-31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of predefined auto loopback frames to be sent. The predefined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload ... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

49.6.2.6 Rate Change Delay Control Register (RCDCR)

Figure 805. Rate Change Delay Control Register (RCDCR)

Offset: 0014h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DRCNT ⁽¹⁾			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 1.

Table 839. RCDCR field description

Bits	Description
0-11	Reserved
12-15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. 0h 0 cycles 1h 1 cycles ... Eh 14 cycles Fh 15 cycles
16-31	Reserved

49.6.2.7 Wakeup Delay Control Register (SLCR)

Figure 806. Wakeup Delay Control Register (SLCR)

Offset: 0018h									Access: User read/write							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HSCNT ⁽¹⁾								0	0	0	0	LSCNT ⁽¹⁾			
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HWKCNT ⁽¹⁾								0	0	0	0	LWKCNT ⁽¹⁾			
W																
Reset	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

1. Only writable when MCR[DRFEN] = 0.

1. Only writable when MCR[DRFEN] = 0.

Table 840. SLCR field descriptions

Bits	Description
0-7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. 00h 0 cycle 01h 1 cycle ... 12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8-11	Reserved
0-15	Reserved
12-15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16-23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. 00h0 cycles 01h1 cycle ... 5Fh95 cycles (1.18 us) ... FFh255 cycles Maximum
24-27	Reserved
28-31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. 0h0 cycle 1h1 cycle 2h2 cycles (1.18 us) ... Eh14 cycles Fh15 cycles

49.6.2.8 ICLC Control Register (ICR)

Figure 807. ICLC Control Register (ICR)

Offset: 001Ch								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ICLSEQ	SNDICLC ⁽²⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	ICLCPLD ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when ICR[SNDICLC] = 0
2. Set by user software and cleared by system hardware.

Table 841. ICR field descriptions

Field	Description
0-13	Reserved
14 ICLCSEQ	ICLC enabled. This bit should be set whenever the S/W is performing a series of ICLC frame transfers. This bit ensures only ICLC frame transmission request is serviced, other pending frames request are ignored. 1 S/W is performing ICLC frame transfers. Only the ICLC frames will be scheduled during this period. All the other frames will be scheduled only after ICR[ICLCSEQ] = 0. 0 Single ICLC frame transfer
15 SNDICLC	ICLC frame request. This bit is set to initiate the transfer of ICLC frame by LFAST master. This bit should be set (ICR[SNDICLC] = 1) after writing the required ICLC payload to be transmitted in the ICLCPLD field of the register. This bit is self clearing, which will be cleared when ICLC frame transfer is complete. 1 Valid ICLC frame for transfer. 0 No Valid ICLC frame for transfer.
16-23	Reserved
24-31 ICLCPLD	ICLC Payload. This field is used to program the payload of the ICLC frame to be transmitted. New ICLC payload should be set when ICR[SNDICLC] = 0. Refer to Table 867 for supported ICLC payloads.

49.6.2.9 Ping Control Register (PICR)

Figure 808. Ping Control Register (PICR)

Offset: 0020h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PNGREQ ⁽²⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PNGAUTO ⁽¹⁾	0	0	0	0	0	0	0	PNGPYLD ⁽¹⁾							
W																
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

1. Only writable when MCR[DRFEN] = 0.

2. Set by user software and cleared by system hardware.

Table 842. PICR field descriptions

Bits	Description
0-14	Reserved
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master 0 Ping response should not be automatically sent. 1 Ping response should be automatically sent.
17-23	Reserved
24-31 PNGPYLD	LFAST Slave: Defines the LFAST slaves ping reply frame payload content. LFAST Master: Defines the expected payload of ping response frame. Used for comparison with ping frame received.

49.6.2.10 Rx FIFO CTS Control Register (RFCR)

Figure 809. Rx FIFO CTS Control Register (RFCR)

Offset: 002Ch Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	RCTSMX ⁽¹⁾					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	RCTSMN ⁽¹⁾					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

1. Only writable when MCR[DRFEN] = 0.

Table 843. RFCR field descriptions

Bits	Description
0-9	Reserved
10-15 RCTSMX	Rx FIFO Maximum Threshold. Defines the condition for CTS bit of frames to be negated.
16-25	Reserved
26-31 RCTSMN	Rx FIFO Minimum Threshold. Defines the condition for CTS bit of frames to be set.

49.6.2.11 Tx Interrupt Enable Register (TIER)

Figure 810. Tx Interrupt Enable Register (TIER)

Offset: 0030h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXIE	TXOVIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGIE	0	TXUNSIE	TXICLCIE	TXDTIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 844. TIER field descriptions

Bits	Description
0-13	Reserved
14 TXIIE	Tx Data Interface Not Enabled — (Mask) Enables or disables the interrupt. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt is disabled 1 Interrupt is enabled
15 TXOVIE	Transmit Data FIFO Overflow Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
16-26	Reserved
27 TXPNGIE	Ping Response Frame Transmitted Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28	Reserved
29 TXUNSIE	Unsolicited Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
30 TXICLCIE	ICLC Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
31 TXDTIE	Data Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled

49.6.2.12 Rx Interrupt Enable Register (RIER)

Figure 811. Rx Interrupt Enable Register (RIER)

Offset: 0034h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	RXUOIE	RXMNIE	RXXMIE	RXUFIE	RXOFIE	RXSZIE	RXICIE	RXLCEIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXCTSIE	RXDIE	RXUNSIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 845. RIER field descriptions

Bits	Description
0-7	Reserved
8 RXUOIE	Unsolicited frame register overflow. Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt is disabled 1 Interrupt is enabled
9 RXMNIE	Rx Data FIFO Min Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
10 RXXMIE	Rx Data FIFO Max Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
11 RXUFIE	Rx Data FIFO Underflow 0 Interrupt is disabled 1 Interrupt is enabled
12 RXOFIE	Rx Data FIFO Overflow 0 Interrupt is disabled 1 Interrupt is enabled
13 RXSZIE	Frame with unsupported frame size received. Valid frame sizes are defined in Table 871 . 0 Interrupt is disabled 1 Interrupt is enabled
14 RXICIE	Invalid ICLC code Received 0 Interrupt is disabled 1 Interrupt is enabled

Table 845. RIER field descriptions (continued)

Bits	Description
15 RXLCEIE	Invalid Logical Channel Type 0 Interrupt is disabled 1 Interrupt is enabled
16-27	Reserved
28 RXCTSIE	Frame with CTS bit Low Received 0 Interrupt is disabled 1 Interrupt is enabled
29 RXDIE	Data frame received 0 Interrupt is disabled 1 Interrupt is enabled
30 RXUNSIE	Unsolicited Frame received 0 Interrupt is disabled 1 Interrupt is enabled
31	Reserved

49.6.2.13 Rx ICLC Interrupt Enable Register (RIIER)

Figure 812. Rx ICLC Interrupt Enable Register (RIIER)

Offset: 0038h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ICPFIE	ICPSIE	ICPRIE	ICTOIE	ICLPIE	ICCTIE	ICTDIE	ICTEIE	ICRFIE	ICRSIE	ICTFIE	ICTSIE	ICPOFIE	ICPONIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 846. RIIER field descriptions

Bits	Description
0-17	Reserved
18 ICPFIE	Ping Frame Response failed 0 Interrupt is disabled 1 Interrupt is enabled

Table 846. RIIE field descriptions (continued)

Bits	Description
19 ICPSIE	Ping Frame Response successful 0 Interrupt is disabled 1 Interrupt is enabled
20 ICPRIIE	ICLC frame for Ping Frame Request received 0 Interrupt is disabled 1 Interrupt is enabled
21 ICTOIE	ICLC frame for Test mode off received 0 Interrupt is disabled 1 Interrupt is enabled
22 ICLPIE	ICLC frame for Loopback On received 0 Interrupt is disabled 1 Interrupt is enabled
23 ICCTIE	ICLC frame for Clk Test mode on received 0 Interrupt is disabled 1 Interrupt is enabled
24 ICTDIE	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt is disabled 1 Interrupt is enabled
25 ICTEIE	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt is disabled 1 Interrupt is enabled
26 ICRFIE	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
27 ICRSIE	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
28 ICTFIE	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
29 ICTSIE	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
30 ICPOFIE	ICLC frame for PLL OFF received 0 Interrupt is disabled 1 Interrupt is enabled
31 ICPONIE	ICLC frame for PLL ON received 0 Interrupt is disabled 1 Interrupt is enabled

49.6.2.14 PLL Control Register (PLLCR)

Figure 813. PLL Control Register (PLLCR)

Offset: 003Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPTMOD			0	0	0	0	0	0	0	0	0	0	0	SWPOFF ⁽²⁾	SWPON ⁽²⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	REFINV	LPCFG		0	0	PLCKCW		FDIVEN	FBDIV ⁽¹⁾								PREDIV ⁽¹⁾	
W																		
Reset	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0		

1. Only writable when MCR[DRFEN] = 0.

2. Set by user software and cleared by system hardware.

Table 847. PLLCR field descriptions

Bits	Description
0-2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 010 Force Vctrl 011 Charge Pump Up 1 100 Charge Pump Up 2 (Internal Resistor) 101 Charge Pump Idle 110 Charge Pump Down 111 Not Used
3-13	Reserved
14 SWPOFF	SW signal to turn OFF the PLL. 0 No effect 1 PLL will be turned OFF.
15 SWPON	SW signal to turn ON the PLL. 0 No effect 1 PLL will be turned ON
16 REFINV	Invert PFD reference clock to avoid noise to other PLLs. 0 Not inverted 1 Invert

Table 847. PLLCR field descriptions (continued)

Bits	Description
17-18 LPCFG	PLL Loop Optimization for sysclk frequency 00 1x multiplier for charge pump current 01 0.5x multiplier for charge pump current 10 1.5x multiplier for charge pump current 11 2x multiplier for charge pump current
19-20	Reserved
21-22 PLCKCW	PLL Lock Ready Count Width. Defines the number of cycles PLL waits for before asserting lock_ready flag High 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider. Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24-29 FBDIV	Feedback Division factor for VCO output clock. 00h No clock output 01h Divide by 2 (1.5, if FDIVEN = 1) 02h Divide by 3 (2.5, if FDIVEN = 1) ... 17h Divide by 24 (23.5, if FDIVEN = 1) 18h Divide by 25 (24.5, if FDIVEN = 1) ... 3Fh Divide by 64 (63.5, if FDIVEN=1)
30-31 PREDIV	Division factor for PLL Reference Clock input. 00b Direct clock passed 01b Divide by 2 10b Divide by 3 11b Divide by 4

49.6.2.15 LVDS Control Register (LCR)

Figure 814. LVDS Control Register (LCR)

Offset: 0040h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SWWKLD ¹	SWSLPLD ¹	SWWKLR ¹	SWSLPLR ¹	SWOFFLD ¹	SWONLD ¹	SWOFFLR ¹	SWONLR ¹
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0	0	0	0	0	LVRXOP			LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

1. Set by user software and cleared by system hardware.

Table 848. LCR field description

Bits	Description
0-7	Reserved
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be put in sleep
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be turned OFF(provided no other source is trying to turn the LD ON)
13 SWONLD	SW signal to turn ON the LVDS LD. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be turned ON
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be turned OFF (provided no other source is trying to turn the LR ON)

Table 848. LCR field description (continued)

Bits	Description
15 SWONLR	SW signal to turn ON the LVDS LR. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be turned ON
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable. The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21	Reserved
22-24 LVRXOP	LVRXOP[2] Termination resistance 0 Disable termination 1 Enable termination LVRXOP[1] Not used LVRXOP[0] Defines the LVDS LR Bias current reference
25-28	Reserved
29 LVTXOP	Control signal for LFAST and Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Data select. This bit is used for selecting use of clock sampled data or normal data in LVDS. 0 normal data to be used inside the LVDS 1 synchronised data to be used inside the LVDS
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

49.6.2.16 Unsolicited Tx Control Register (UNSTCR)

Figure 815. Unsolicited Tx Control Register (UNSTCR)

Offset: 0044h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																USNDRQ ⁽²⁾
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	UNSHDR ⁽¹⁾						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0
2. Set by user software and cleared by system hardware.

Table 849. UNSTCR field descriptions

Bits	Description
0-14	Reserved
15 USNDRQ	Tx Unsolicited send request. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16-24	Reserved
25-31 UNSHDR	Tx Unsolicited message header.

49.6.2.17 LFAST Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)

Figure 816. Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)

Offset: 0048h (UNSTDR8)...0068h (UNSTDR0)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UNTXD[31:16] ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNTXD[15:0] ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0

Table 850. UNSTDR_n field descriptions

Bits	Description
0-31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

49.6.2.18 Global Status Register (GSR)

Figure 817. Global Status Register (GSR)

Offset: 0080h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0	0	0	0	0	0	0	0	0	0	0	0	LRMD	LDSD	DRSD
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LPTXDN	LPFPDV	LPCLDV	LPCHDV	LPCLDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 851. GSR field descriptions

Bits	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module Slave only mode 1 LFAST Module in dual mode
1-12	Reserved
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active. 0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Recieve Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16-26	Reserved
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by Auto Loopback Control Register (ALCR) LPFMCNT. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

49.6.2.19 Ping Status Register (PISR)

Figure 818. Ping Status Register (PISR)

Offset: 0084h Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RXPNGD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 852. PISR field descriptions

Bits	Description
0-23	Reserved
24-31 RXPNGD	Ping Data Register. In LFAST Master mode the Ping response ICLC frame received is stored into this register.

49.6.2.20 Data frame Status Register (DFSR)

Figure 819. Data frame Status Register (DFSR)

Offset: 0094h Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXDCNT						0	0	0	0	0	RXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			TXDCNT						0	0	0	0	0	TXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 853. DFSR field descriptions

Bits	Description
0-1	Reserved
2-7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8-12	Reserved
13-15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
18-23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24-28	Reserved
29-31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

49.6.2.21 Tx Interrupt Status Register (TISR)

Figure 820. Tx Interrupt Status Register (TISR)

Offset: 0098h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 854. TISR field descriptions

Bits	Description
0-13	Reserved
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table 854. TISR field descriptions (continued)

Bits	Description
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16-26	Reserved
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28	Reserved
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

49.6.2.22 Rx Interrupt Status Register (RISR)

Figure 821. Rx Interrupt Status Register (RISR)

Offset: 009Ch								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 855. RISR field descriptions

Bits	Description
0-7	Reserved
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. 0 Interrupt event has not occurred 1 Interrupt event has occurred- On reception of frame with payload size for a frame other than mentioned in Table 871 .
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in Table 871 .
16-27	Reserved
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
31	Reserved

49.6.2.23 Rx ICLC Interrupt Status Register (RIISR)

Figure 822. Rx ICLC Interrupt Status Register (RIISR)

Offset: 00A0h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONE
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 856. RIISR field descriptions

Bits	Description
0-17	Reserved
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table 856. RIISR field descriptions (continued)

Bits	Description
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

49.6.2.24 PLL and LVDS Status Register (PLLLSR)

Figure 823. PLL and LVDS Status Register (PLLLSR)

Offset: 00A4h								Access: User read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PLDIS	PLDCR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LRSLPS	LDSLPS	LDPDS	LRPDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 857. PLLSR field descriptions

Bits	Description
0-13	Reserved
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles. 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0
16-27	Reserved
28 LRSLPS	This bit indicates the real time status of the LR sleep signal. 0 LR sleep signal is negated. 1 LR sleep signal is asserted.
29 LDLPS	This bit indicates the real time status of LD sleep signal. 0 LD power sleep signal is negated. 1 LD power sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LR power down signal. When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LD power down signal. When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.

49.6.2.25 Unsolicited Rx Status Register (UNSRSR)

Figure 824. Unsolicited Rx Status Register (UNSRSR)

Offset: 00A8h

Access: read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	URXDV	0	0	0	0	0	URPCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 858. UNSRSR field descriptions

Bits	Description
0-22	Reserved
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24-28	Reserved
29-31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited data payload present in the Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0). The bitfield coding is same as b7-b5 in Table 831 .

49.6.2.26 Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Figure 825. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

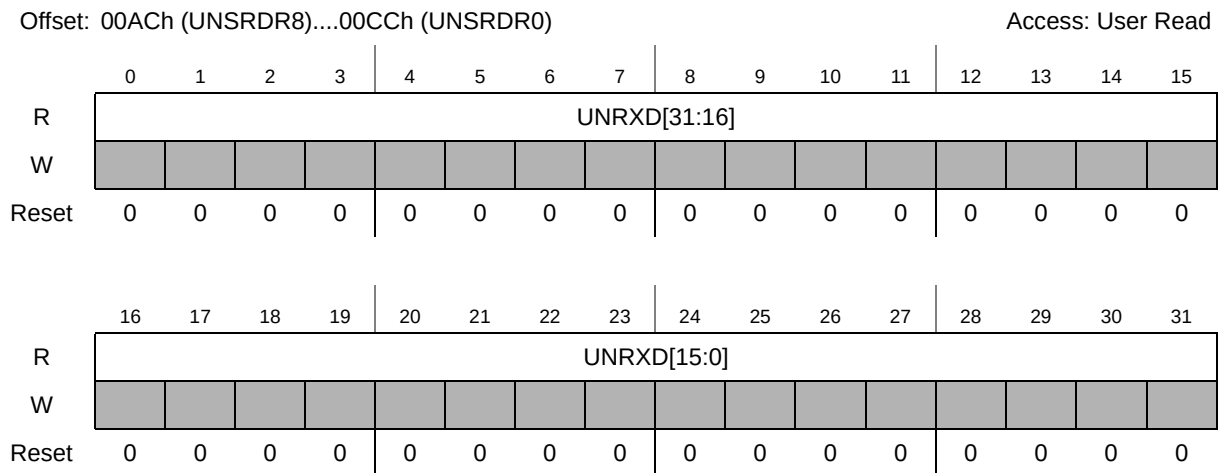


Table 859. UNSRDRn field descriptions

Bits	Description
0-31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data. It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

49.6.3 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements⁽ⁿ⁾. All registers of the LFAST module are classified as non-safety relevant.

The MCR[IPGDBG] fulfills the safety requirements for monitoring of the debug freeze signal.

n. See the "Functional Safety" chapter for register classification details.

49.7 Functional description

49.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

49.7.1.1 LFAST master interface startup procedure

1. After reset the SLCR and RCDCCR are programed according to the LVDS parameters in the device datasheet.
2. The PLLCR is programmed with configuration parameters for the PLL.
3. The LCR is programed with configuration parameters of the LVDS.
4. Write MCR[MSEN] = 1. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. Write MCR[RXEN] = 1 and MCR[TXEN] = 1 to negate the LD powerdown, LR disable and LR powerdown signals.
7. Write ICR[ICLCPLD] = 31h to enable the Slaves Tx Interface.
8. Write ICR[SNDICLC] = 1 and ICR[ICLCSEQ] = 1.
9. Write MCR[TXARBD] = 0.
10. The ICLC transmission is confirmed by verifying one of the following:
 - ICR[SNDICLC] = 0.
 - TISR[TXICLCF] = 1.
11. Write ICR[ICLCPLD] = 00h to check the LFAST slaves status.
12. Write ICR[SNDICLC] = 1.
13. The LFAST slave status is confirmed by occurence of one of the following:
 - LFAST slave is enabled if RIISR[ICPSF] = 1. Proceed to step 14.
 - LFAST slave is disabled if RIISR[ICPFF] = 1. The LFAST master must wait and restart from Step 7.

Speed mode change:

14. Write LCR[SWPON] = 1 to enable the LFAST masters PLL.
15. Write ICLC start PLL frame, ICR[ICLCPLD] = 02h.
16. Write ICR[SNDICLC] = 1.
17. The ICLC transmission is confirmed by occurrence of one of the following:
 - ICR[SNDICLC] = 0.
 - TISR[TXICLCF] = 1.
18. To change LFAST masters Tx interface speed both of the following operations are performed:
 (Note: The slaves Rx interface speed should be changed first.)
 - Write ICR[ICLCPLD] = 10h for Tx data fast frame.
 - Write ICR[SNDICLC] = 1.
19. Both of the following operations are performed to change the LFAST masters Rx interface speed:
 (Note: The slaves Tx interface speed mode should be changed first)
 - Write ICR[ICLCPLD] = 80h to select Rx data fast frame.
 - Write ICR[SNDICLC] = 1.
20. The ICLC transmission is confirmed by the occurrence of one of the following:
 - ICR[SNDICLC] = 0
 - TISR[TXICLCF] = 1.
21. Write SCR[TDR] = 1 to change the LFAST masters Tx interface speed.
22. Write SCR[RDR] = 1 to change the LFAST masters Rx interface speed.
23. Write ICR[ICLCPLD] = 00h to confirm the change in speed of the LFAST slave. This frame should be written after waiting for the expected delay in the start of the PLL and speed mode change delay at the LFAST slave.
24. Write ICR[SNDICLC] = 1
25. Write MCR[TXARBD] = 1, after some delay to ensure the step 24 ICLC frame is send but no other frame is sent to arbitration.
26. The LFAST slaves speed mode is confirmed by the occurrence of the following:
 - If RIISR[ICPSF] = 1. The LFAST slave is in High Speed mode.
 - If RIISR[ICPFF] = 1. The LFAST slave is in Low Speed mode.
27. If RIISR[ICPSF] = 1, then all frame arbitration is enabled by both of the following operations:
 - Write ICR[ICLCSEQ] = 0.
 - Write MCR[TXARBD] = 0.
28. If RIISR[ICPFF] = 1 modifying the LFAST slaves speed needs to be repeated by:
 - Write SCR[TDR] = 0 to return the Tx interface to Low Speed mode.
 - Write MCR[TXARBD] = 0, to enable frame transmission.
 - The operations from step 14 need to be executed again.

49.7.1.2 LFAST slave interface startup procedure

1. After reset the SLCR and RCDCCR are programed according to the LVDS parameters in the device datasheet.
2. The PLLCR is programed with the configuration parameters for the PLL.
3. The LCR is programed with the configuration parameters for the LVDS.
4. Write MCR[MSEN] = 0. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. The LR is enabled by writing MCR[RXEN] = 1. This negates the LR disable signal and LR's power down signal.
7. The LD enable signal needs to be asserted. This is done when one of the following are true:
 - When an ICLC frame with payload 31h is received the H/W will write RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
 - MCRMCR[TXEN] = 1
8. After a write to MCR[TXARBD] = 0, a ping frame will be sent on one of the following conditions:
 - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h frame is received.H/W writes [ICRPF] = 1
 - PICR[PNGREQ] = 1.

Speed mode change:

9. The PLL will start when one of the following conditions is met:
 - When an ICLC frame with payload 02h is received. H/W writes RIISR[IPONF] = 1
 - Write PLLCR[SWPON] = 1.
10. The speed of the Tx interface is changed on one of the following conditions:
 - When SCR[DRMD] = 1 and an ICLC frame with payload 80h is received. H/W writes RIISR[ICTFF] = 1 and SCR[TDR] = 1.
 - SCR[TDR] = 1 when SCR[DRMD] = 0
11. The speed of the Rx interface is changed on one of the following conditions:
 - When SCR[DRMD] = 1 and an ICLC frame with payload 10h is received. H/W writes RIISR[ICRFF] = 1 and SCR[RDR] = 1.
 - SCR[RDR] = 1, when SCR[DRMD] = 0
12. A Ping Frame is sent on one of the following conditions:
 - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h is received.H/W writes RIISR[ICRPF] = 1
 - Write PICR[PNGREQ] = 1.

49.7.2 Line Receiver (LR)

49.7.2.1 Introduction

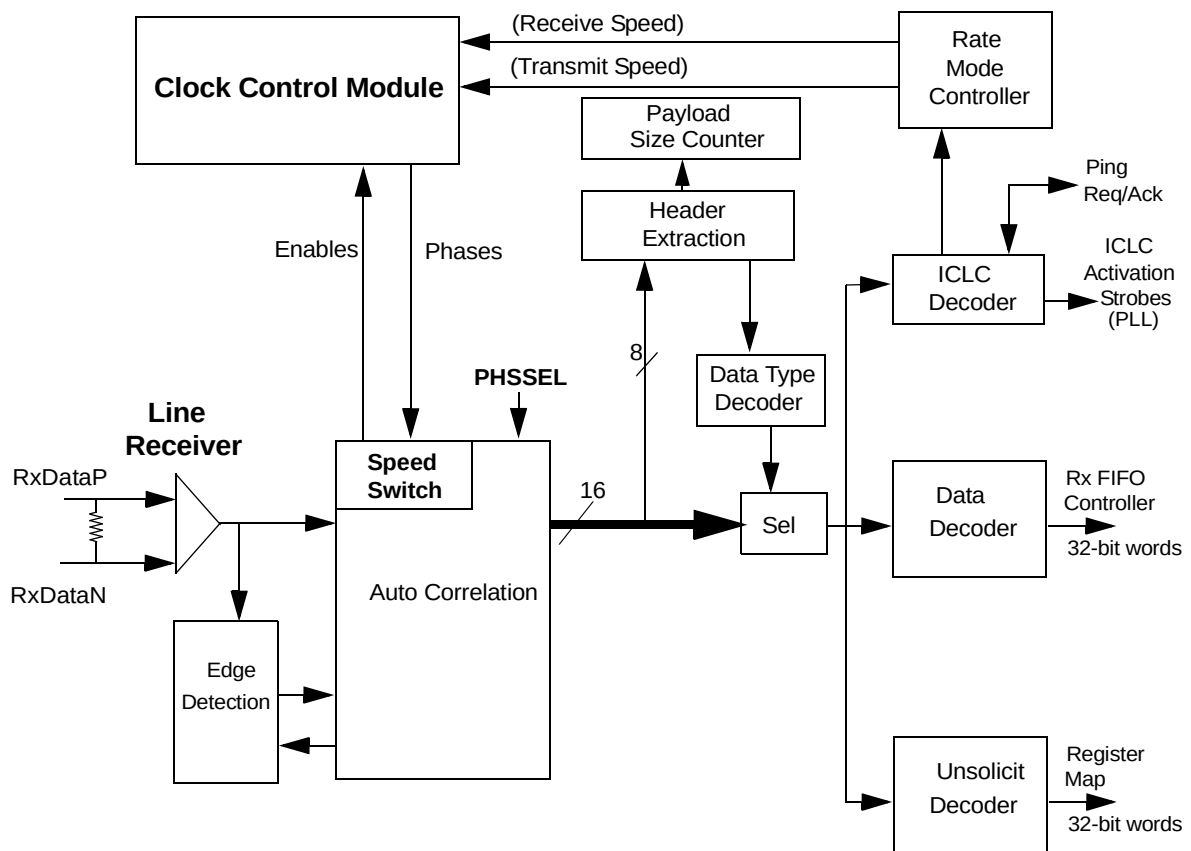
The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling

edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 826 shows the block diagram of Uplink Controller.

Figure 826. Top Level Receive Controller



49.7.2.2 Edge-Detection & Auto correlation

The edge-detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge-detection circuitry is used or not. The edge-detection will be performed first if required before auto-correlation.

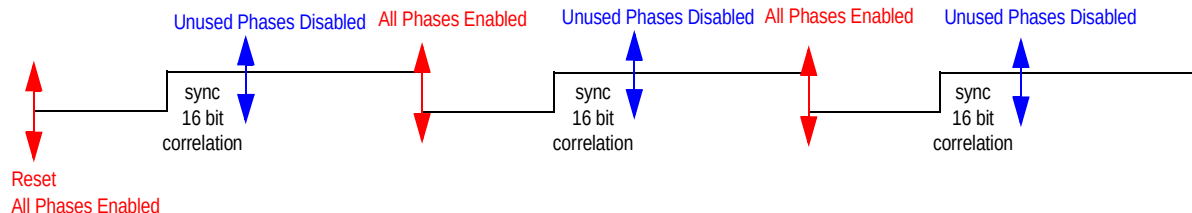
49.7.2.2.1 Auto correlation modes

49.7.2.2.1.1 Hunt correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the

synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 827. Hunt Correlation mode



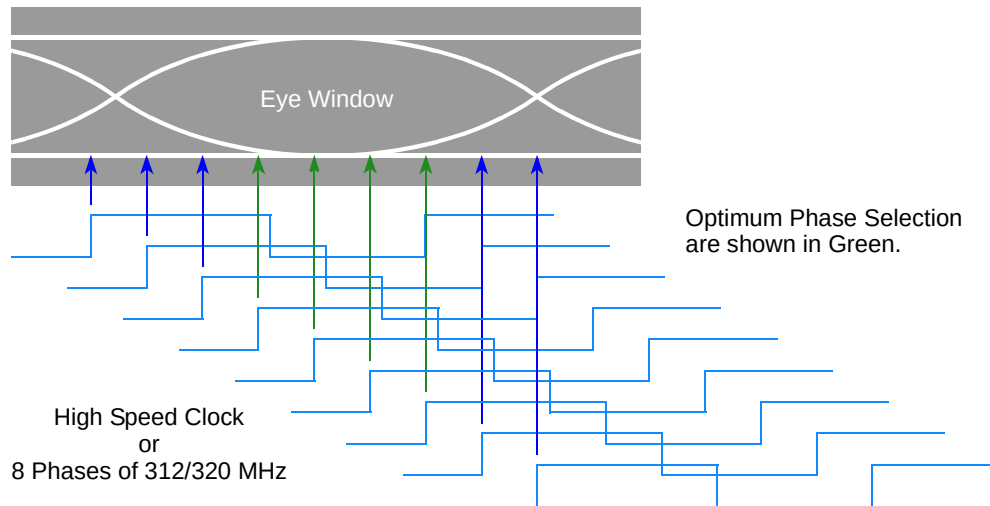
49.7.2.2.2 Edge detection

The edge detector is disabled in Hunt Correlation mode.

49.7.2.2.3 Auto correlation

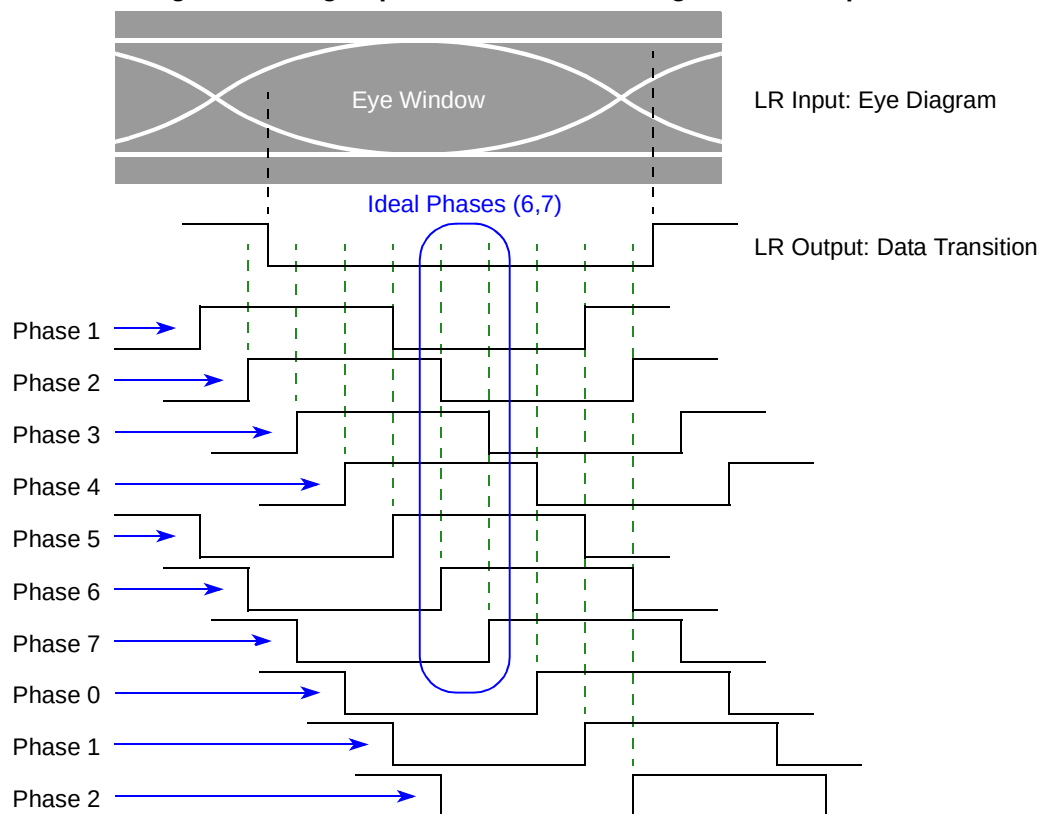
The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in [Figure 828](#).

Figure 828. High Speed 8 Phase Selection



The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in [Figure 829](#).

Figure 829. High Speed 8 Phase Clock Alignment Example



Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in [Figure 830](#) and [Figure 831](#).

Figure 830. High Speed 4 Phase Clock Alignment Example

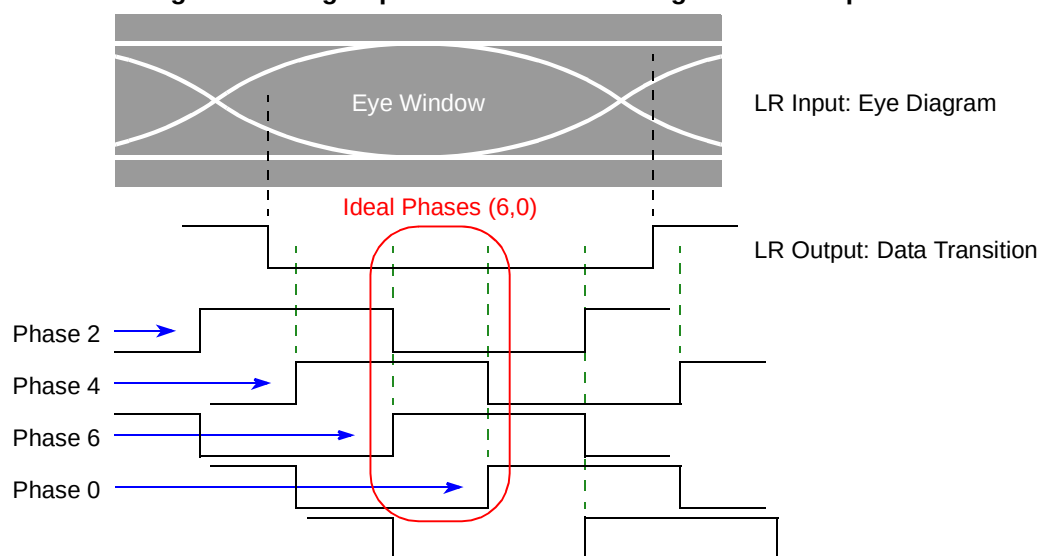
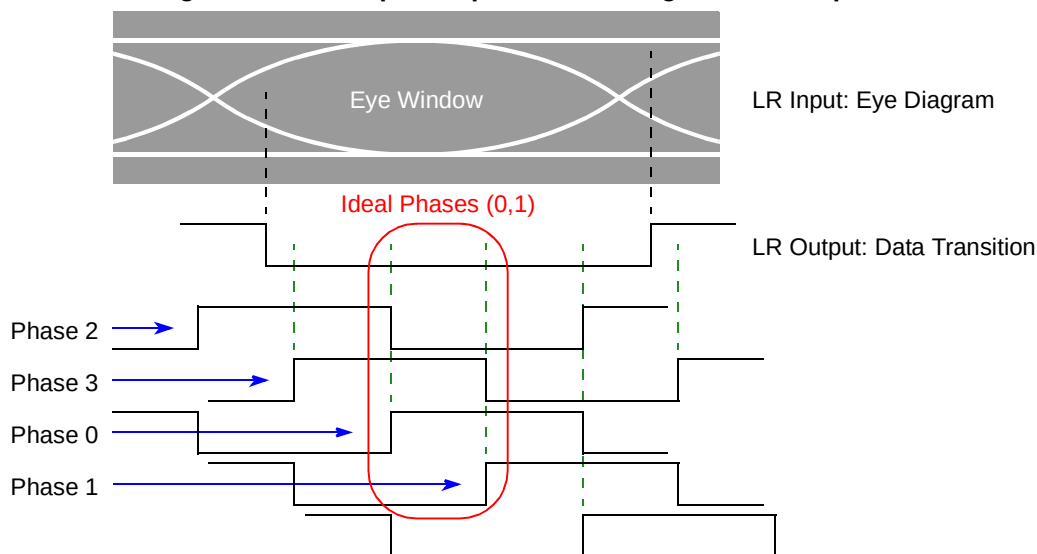


Figure 831. Low speed 4 phase clock alignment example



The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed (312/320 MHz) 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High Speed 8 Phases: Samplers 0,1,2,3,4,5,6,7
- High Speed 4 Phases: Samplers 0,2,4,6
- Low Speed 4 Phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and re-samples it to an intermediate phase as shown in [Table 860](#), [Table 861](#), and [Table 862](#) before re-sampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 860. High speed 8 phase selection - sampling procedure

Samplers	Initial Sample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 861. High speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See [Table 861](#), Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 862. Low speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See [Table 862](#), Samplers 4, 5, 6, 7 are disabled. The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

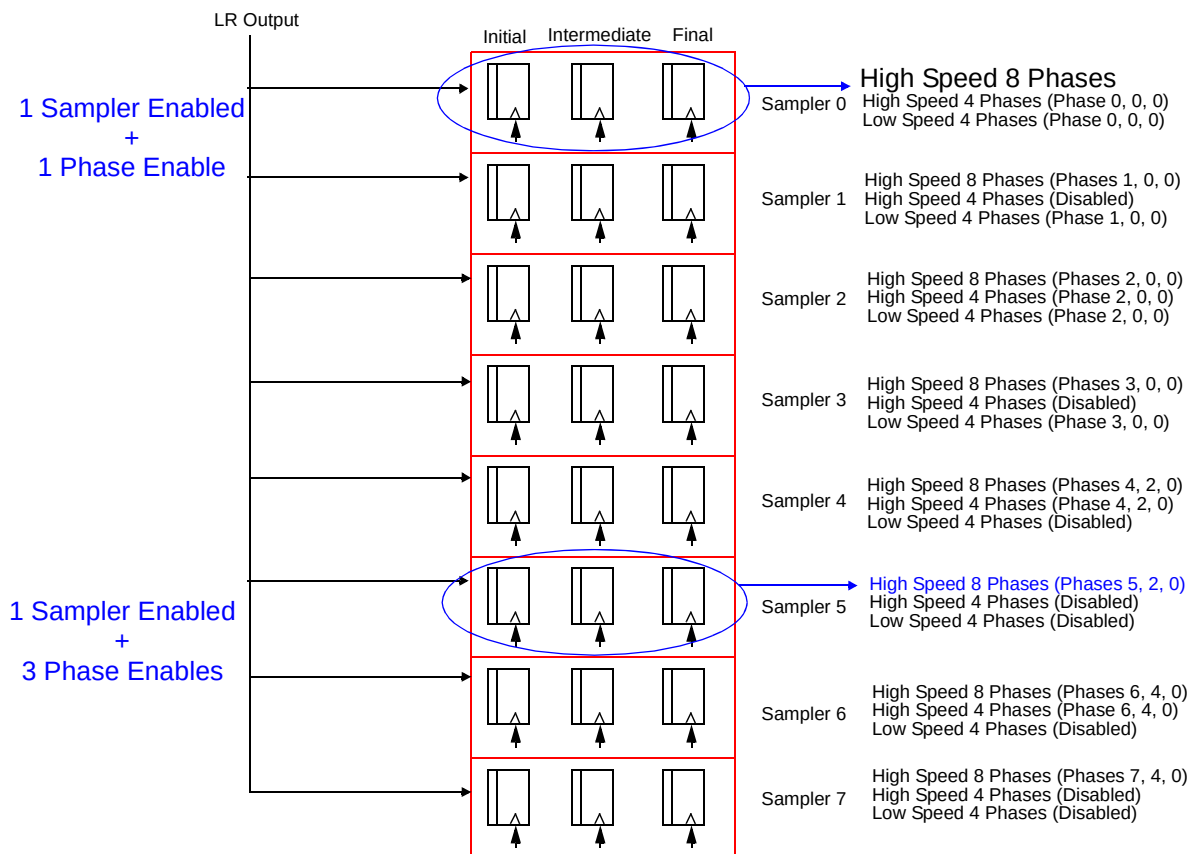
49.7.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in [Figure 832](#).

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (e.g Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example, Sampler 0 has all Phase 0 content).

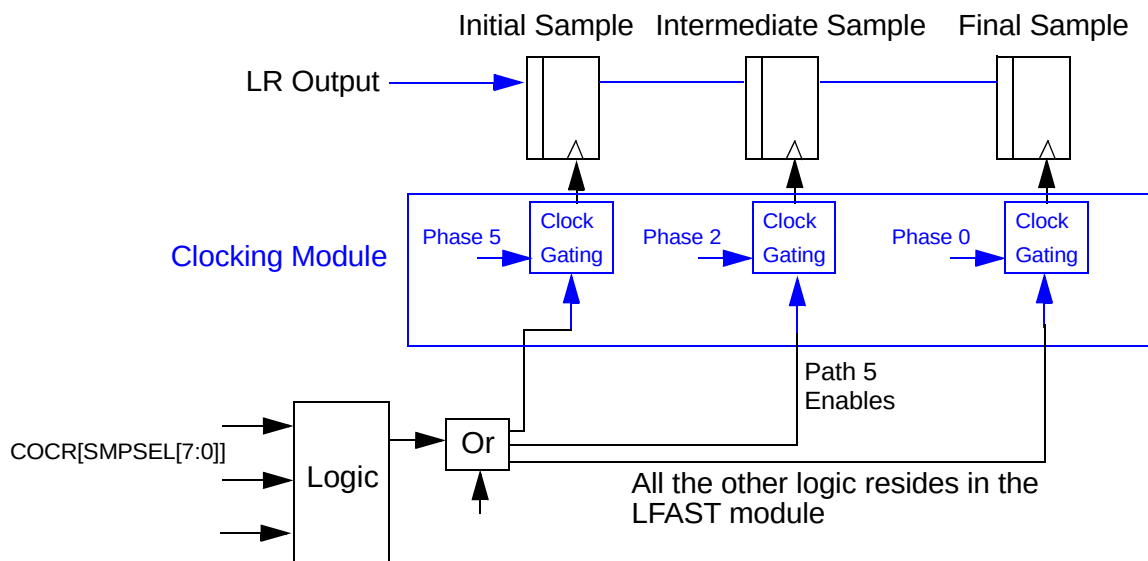
The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by de-asserting the respective phase enables.

Figure 832. Samplers, each Sampler has 3 Registers



In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in [Figure 833](#).

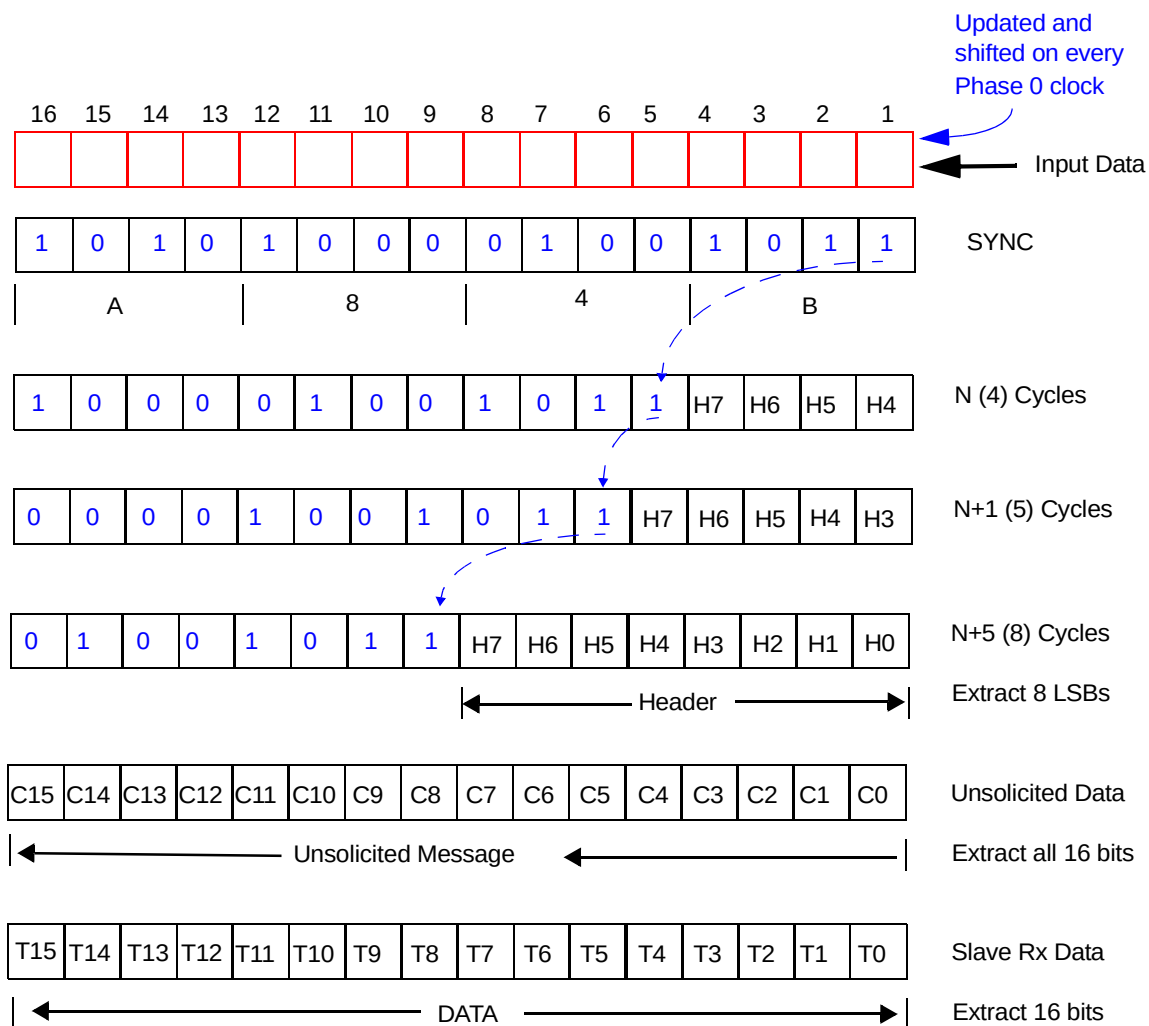
The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

Figure 833. Sampler 5 Logic

49.7.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in [Figure 834](#)

Figure 834. Extracting Header and Payload from the 16-bit output from Auto correlation.



49.7.2.3.1 Line Receiver states

The LD has the following states:

- Shutdown**
The LR enters in the shutdown state when the power down signal is high. In this state the LR regulator is supplying a Vdd level, however the LR is not enabled. In this state the LR holds its off value. Once power down signal is negated, a predefined settling time is required before the LR may be used for communication. During the settling time the peer LFAST device should not start its transfer.
- Sleep**
The LR enters in the Sleep state when the sleep signal is high. In the sleep state, the LR is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In this

mode the receiver current consumption is reduced. Hysteresis in the LR ensures that it continues to present a 1 to the LFAST module.

- **Normal**
In the Normal state, the LR is primed for reception. Receiver converts differential signals to a single ended logic signal.

Table 863. Line Receiver State

LR State	LFAST Master Triggers (at least one of the following)	LFAST Slave Triggers (at least one of the following)
Shutdown	– Programing of LCR[SWOFFLR] or LCR{SWONLR] – Programing MCR[RXEN] and MCR[TXEN]	– LFAST interface enable assertion/negation – Programing of LCR[SWOFFLR] or LCR{SWONLR] – Programing MCR[RXEN] and MCR[TXEN]
Sleep	– Programing of LCR[SWSLPLR] or LCR[SWWKLR]	– Programing of LCR[SWSLPLR] or LCR[SWWKLR]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

49.7.3 Transmit Controller

49.7.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, CTS, ping etc. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Recieve Controller and the ping response from the register block.

The Tx interface has two speed modes:

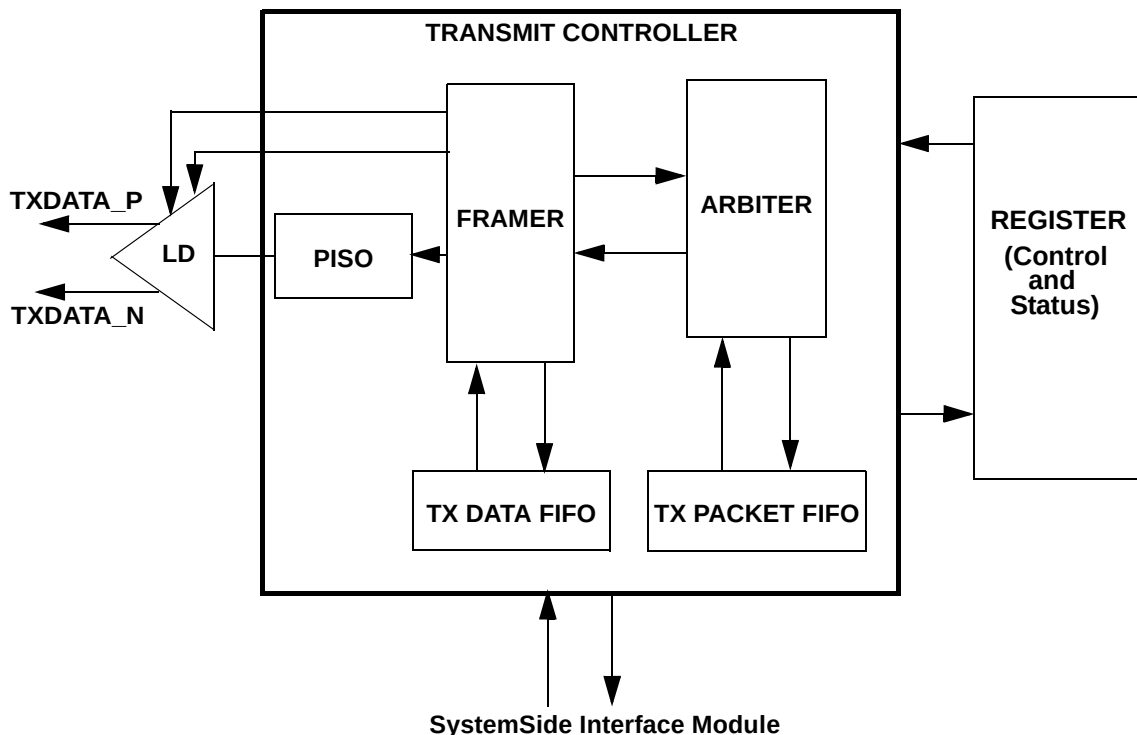
- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in [Figure 835](#).

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

Figure 835. Transmit Controller Connections



49.7.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like:

- S/W programmable registers
- System side interface FIFO
- Rx interface controller

The level of priority for each request is shown in [Table 864](#).

Table 864. Priority Levels for the Transmit Controller

Request	LFAST Master Priority	LFAST Slave Priority	Triggering Conditions (at least one of the listed)
LFAST interface enable	1	1	– LFAST interface enable is asserted – LFAST interface enable is negated
Tx Interface disabled	2	2	– MCR[DRFEN] = 0 – MCR[TXEN] = 0 – MCR[DRFRST] = 1 – MCR[TXARBD] = 1 – LFAST Slave: ICLC frame with payload for “Disable Rx interface” received

Table 864. Priority Levels for the Transmit Controller

Request	LFAST Master Priority	LFAST Slave Priority	Triggering Conditions (at least one of the listed)
Tx Interface speed mode change	3	3	– SCR[TDR] is modified – LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	4	– TMCR[LPON] = 1 – LFAST Slave: ICLC frame with payload for loopback mode enable received <i>Note:</i> Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
ICLC frame request	5	–	– LFAST Master: ICR[SNDICLC] = 1 – LFAST Master: ICR[ICLCSEQ] = 1
Ping response request	–	5	– LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 – LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	6	– Last Frame with CTS = 1 received from the peer LFAST device – UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	7	– Tx FIFO contains one or more frames – Last Frame with CTS = 1 received from the peer LFAST device – MCR[DATAEN] = 1
CTS Frame	8	8	– MCR[CTSEN] = 1 – Rx FIFO reaches High/Low threshold, defined by TISR and no frame pending
Clock mode test	9	9	– TMCR[CLKTST] = 0 – LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCCR[DRCNT].

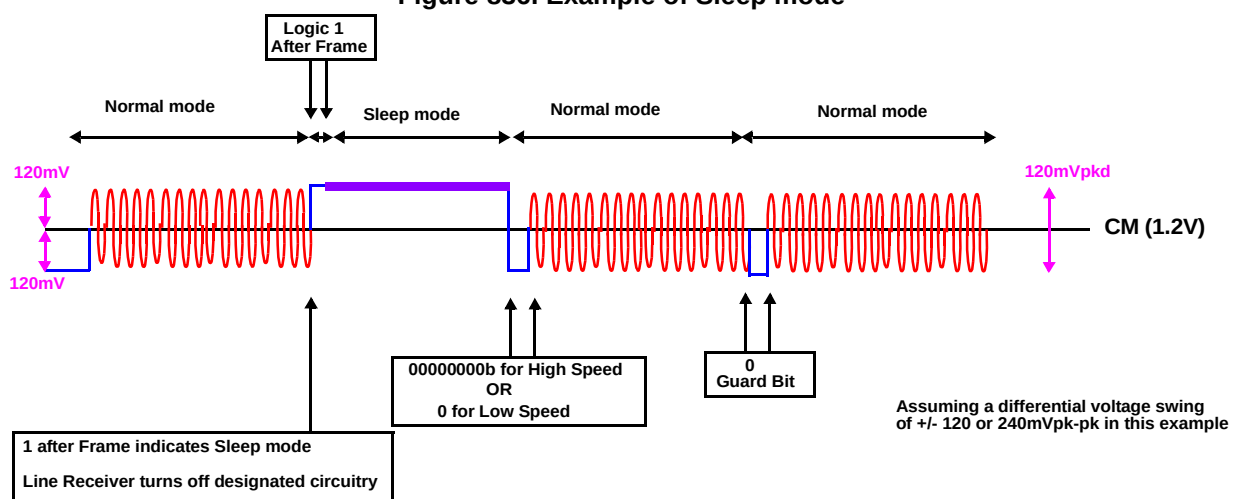
49.7.3.3 Line Driver digital connections

49.7.3.3.1 Line Driver states

The LD has the following states:

- Shutdown**
The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling time is required before the LD may be used for communication. The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.
- Sleep**
The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In the sleep state the LD outputs are connected to V_{cm} (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a predefined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

Figure 836. Example of Sleep mode



As shown in [Figure 836](#) before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal**
In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. If the LD is not enabled, and the framer is activated, and has a frame to transmit, an interrupt (TISR[TXIEF]) will be generated. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 865. Line Driver States

LD State	LFAST Master Triggers	LFAST Slave Triggers
Shutdown	– Programing LVDS[SWOFFLD] and LVDS[SWONLD] – Programing MCR[TXEN] and MCR[DRFEN]	– LFAST interface enable negation/assertion – Programing LVDS[SWOFFLD] and LVDS[SWONLD] – ICLC command from LFAST master – Programing MCR[TXEN] and MCR[DRFEN]
Sleep	– Programing LVDS[SWSLPLD] and LVDS[SWWKLD]	– Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

49.7.4 CTS mode support

LFAST module supports flow control method defined in the LFAST3G standard. The CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by proper programing of the Rx FIFO Lower threshold (RFCR[CTSMN]) and Higher threshold (RFCR[CTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by write MCR[CTSEN] = 1 then:

- Sends CTS bit as 0 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield TISR[CTSMX]. This indicates that the LFAST device does not want the peer device to send data. The CTS bit of all frame are sent 0 until the Rx FIFO pointer reaches the lower threshold, as described below.
- Sends CTS bit as 1 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield TISR[CTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frame are sent 1 until the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of Header field are sent as 0011b and 8-bit payload of 0.

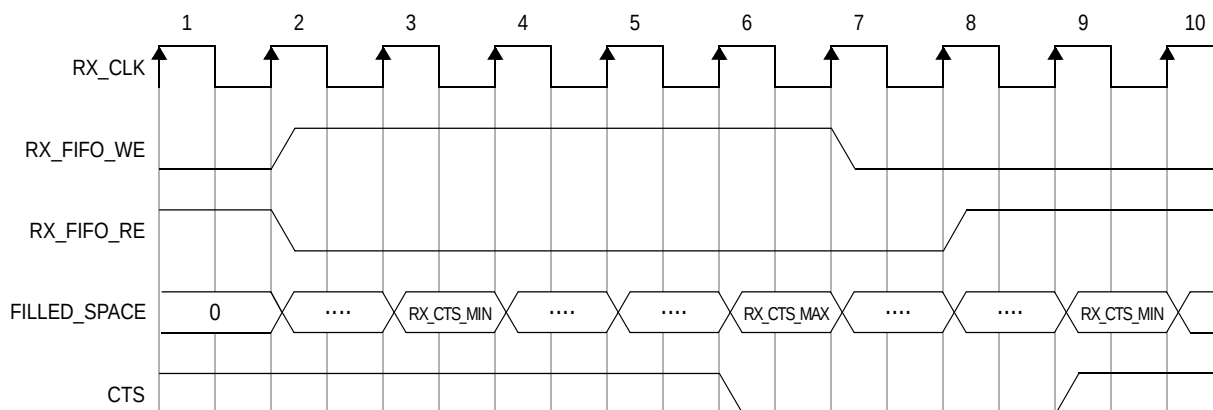
If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frame sent will have CTS bit as 1 in LFAST frame (bit[0] of Header field).

CTS Reception:

The reception of a LFAST frame with CTS bit 0 indicates that the peer device is not ready to receive data frames. The transmission of all pending data and unsolicited frames are postponed until a frame with CTS bit 1 is received.

Figure 837. CTS generation



49.7.5 Frames supported

Table 866 provides the frames supported and their permissible payload sizes

Table 866. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx 2. LFAST Slave Rx	Supported by 1. LFAST Slave Tx 2. LFAST Master Rx	Payload Size (bits)
Data Frame	0100b – 1011b	YES	YES	32, 64, 96, 128, 256, 288
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO ⁽¹⁾	8
CTS Frame	0011b	YES	YES	8
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

If logical channel type (LCT) code other than the above mentioned are received then the interrupt flag RISR[RXLCEF] = 1. If the payload size received does not match any value in Table 866 then the flag RISR[RXSZF] = 1. In both these error conditions the received frame is ignored. The S/W may have to take the necessary steps to recover from such an error.

49.7.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

49.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

49.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO. Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

49.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTE] = 1 after each frame transfer.
5. CTS is set depending on the push/pull mode setting defined by MCR[CTSEN].

49.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[TXEN] and MCR[DRFEN] bits should be set to enable the data frame reception. When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded. In this case, UNSRSR[RXOF] = 1 and an interrupt is generated if the RIER[RXOFIE] = 1.

49.7.6.2 Unsolicited flow**49.7.6.2.1 Unsolicited frame transmit flow**

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

49.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
 - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
 - UNSTCR[USNDRQ] = 0
 - TISR[TXUNSF] = 1.
 - If UNSTCR[USNDRQ] = 0 then:
 - Program the unsolicited payload in UNSTD0–UNSTD8, and can be written up to a payload of frame of a maximum of 288 bits.
 - Program the unsolicited frame header in UNSTCR[UNSHDR].
 - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.
6. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

49.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTD0–UNSTD8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.
3. Then it clears the interrupt (write RISR[RXUNSF] = 1) and also the UNSRSR[URXDV] bit.

49.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is defined as a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

49.7.6.3.1 ICLC data transmit flow

Whenever the processor intends to transfer an ICLC frame to the LFAST slave then the following steps are to be followed.

1. Write the ICLC frame payload in the ICR[ICLCPLD] (see [Table 867](#) for payloads as defined by the standard).
2. Program ICR[SNDICLC] = 1 to initiate the ICLC frame transfer. This bit clears itself when the ICLC frame has been transmitted. The processor needs to poll this bit to make sure it is cleared before setting this bit again (even when ICR[ICLCSEQ] = 1).
3. TISR[TXICLCF] = 1 after the transfer of the ICLC frame.
4. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

To determine whether the ICLC frame has been sent or not, either the ICR[SNDICLC] bit can be polled, or wait for TISR[TXICLCF] = 1.

Table 867. Supported ICLC Payloads

ICLC Code (hex)	ICLC Function
00	Ping Request from LFAST Master to LFAST Slave
01	Reserved
02	Start PLL In preparation for High Speed mode
04	Stop PLL Fallback to low speed mode
08	Select TxData Slow (LFAST Slave Rx Interface)
10	Select TxData Fast (LFAST Slave Rx Interface)
20	Select RxData Slow (LFAST Slave Tx Interface)
40	Not Supported
80	Select RxData Fast (LFAST Slave Tx Interface)
31	Enable RxData Interface (LFAST Slave Tx Interface)
32	Disable RxData Interface (LFAST Slave Tx Interface)
34	Clock Test mode Send 101010... continuously using the currently configured clock rate (slow, fast);
FF	Turn payload loopback on
38	Turn Test mode (Loopback and Clock Test) off

Programing model for ICLC frame transmit:

1. Set the ICR[ICLCSEQ] bit (optional).
2. Write the ICLC frame payload in the ICR[ICLCPLD], corresponding to the desired Tx/Rx data rate change, as described in the [Table 867](#).
3. Write ICR[SNDICLC] = 1 to initiate the ICLC frame transfer.
4. LFAST master will schedule the ICLC transfer. Data present in ICR[ICLCPLD] is transmitted to the LFAST slave at the old Tx data rate. LFAST slave will configure its Tx/Rx interface data rate accordingly.
5. Reset the ICR[ICLCSEQ] to allow the scheduling of other types of frames, which will be sent at the new data rate.

When ICR[ICLCSEQ] = 0, though the ICLC frame transfer still has highest arbitration priority, other frames may be scheduled if no valid ICLC frame request exist.

Note: ICLC frame transmit will not trigger the internal data rate change of LFAST Master Tx/Rx interface. LFAST master Tx/Rx interface data rate will only be changed, when corresponding SCR[TDR] and SCR[RDR] bits are appropriately configured.

Processor requires to take care that no data frames are to be transmitted while data rate change is happening or ICLC is transmitting frames by setting ICR[ICLCSEQ] bit in [Table 867](#).

49.7.6.3.2 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

The reception of an ICLC frame is indicated by an interrupt to the system. The ICLC status register RISR indicates the type of the ICLC frame received. The Rx block will decode and generate the appropriate signals. An invalid ICLC code besides the ones listed in [Table 867](#) will cause RISR[RXICF] = 1.

49.7.6.3.2.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF] = 1. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

49.7.6.3.2.2 Start PLL ICLC

The Start PLL ICLC command will write bit RIISR[ICPONF] = 1.

49.7.6.3.2.3 Stop PLL ICLC

The Stop PLL ICLC command will write RIISR[ICPOFF] = 1.

49.7.6.3.2.4 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

49.7.6.3.2.5 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

49.7.6.3.2.6 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

49.7.6.3.2.7 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

49.7.6.3.2.8 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

49.7.6.3.2.9 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable .

49.7.6.3.2.10 Ping response ICLC

The LFAST master considers any ICLC frame payload as ping response data. The LFAST masters Rx block will store the received ping data in PISR[RXPNGD] and match it with PICR[PNGPYLD]. RIISR[ICPSF] = 1 if the Ping received matches with the one stored in PICR[PNGPYLD], or RIISR[ICPFF] = 1.

49.7.6.4 CTS flow

49.7.6.4.1 CTS Tx flow

The CTS frame, if enabled by the S/W, indicates the status of the Recieve Data FIFO. CTS frame is triggered whenever Rx Data FIFO reaches either the High/Low threshold and no data, ICLC or unsolicited frame is ready for transmission.

Programing model for CTS frame transmit:

- Program RFCR[CTSMX] and RFCR[CTSMN].
- Enable the CTS frame Transmission by programming MCR[CTSEN] = 1.
- Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path.
- The CTS bit sent with a valid frame (for example, data, ICLC and unsolicited) is 1 until the Rx Data FIFO reaches High Threshold.
- When the Rx Data FIFO reaches higher threshold and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 0 and the status bit RISR[RXMXF] = 1.
- When the Rx Data FIFO reaches Low Threshold, due to system side reads and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 1, and status bit RISR[RXMNF] = 1.

49.7.6.4.2 CTS Rx flow

Whenever a CTS frame header or any other frame with valid header is received with CTS bit = 0, then RISR[RXCTSF] = 1. The Tx Interface arbitration for unsolicited frame and data frame transmit request is turned off on reception of frame with CTS bit 0. The arbitration for unsolicited frame and data frame is enabled on reception of frame with CTS bit 1. Frame with LCT type CTS are not stored in the LFAST.

49.7.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

49.7.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programmed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 868. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

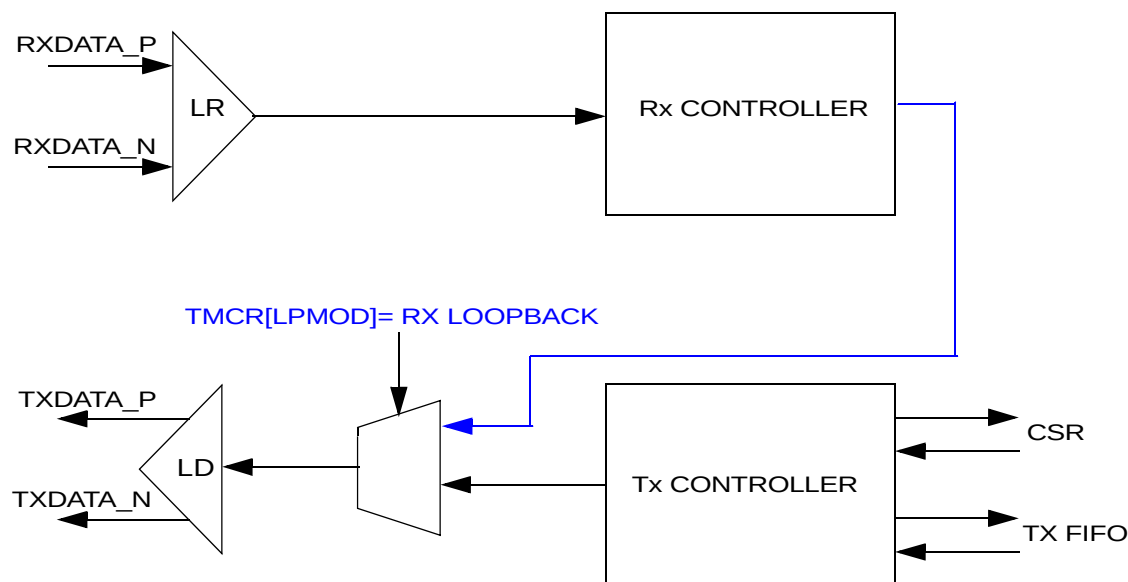
- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

49.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. CTS bit (Bit 0) of the header is guaranteed to be asserted when

the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

Figure 838. Rx LoopBack mode



Entry to Rx Loopback mode:

1. S/W programs `TMCR[LPMOD] = 000b`.
2. Loopback can be turned on by either of the following methods:
 - S/W programs `TMCR[LPON] = 1`.
 - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

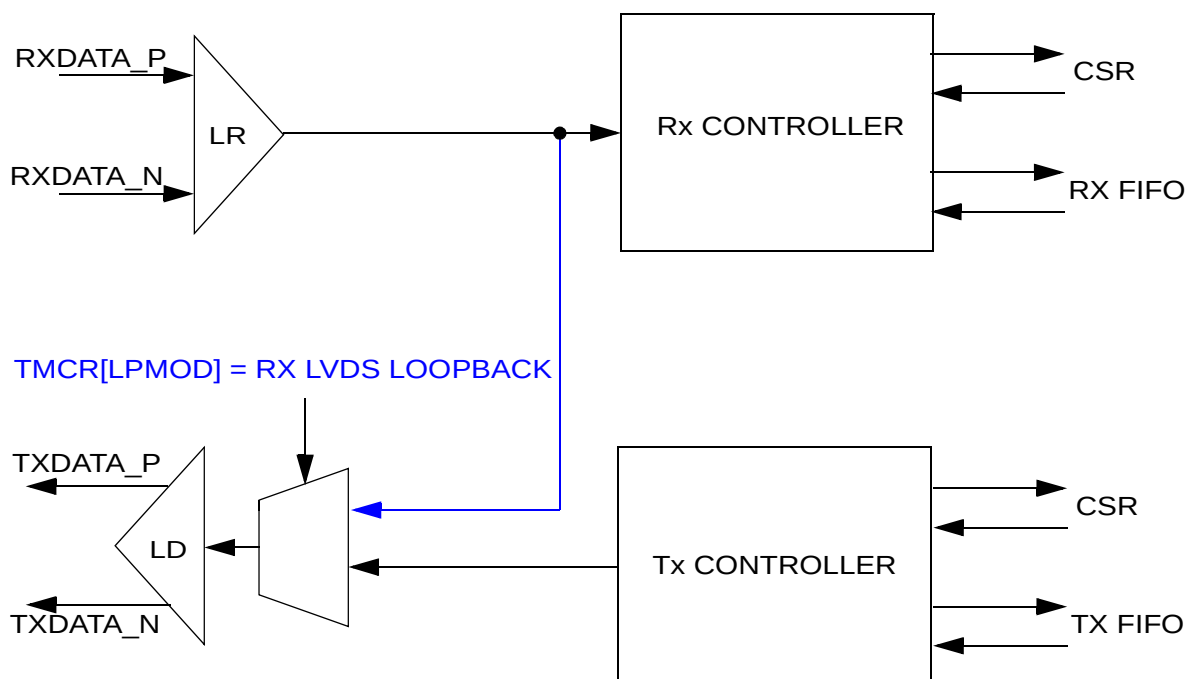
- S/W programs `TMCR[LPON] = 0`
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

49.7.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via “Rx LVDS LoopBack Mux”. Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

Figure 839. Rx LVDS LoopBack mode



Entry to Rx LVDS Loopback mode:

1. S/W programs `TMCR[LPMOD] = 001b`.
2. Loopback can be turned ON by either of the following methods:
 - S/W writes `TMCR[LPON] = 1`.
 - For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

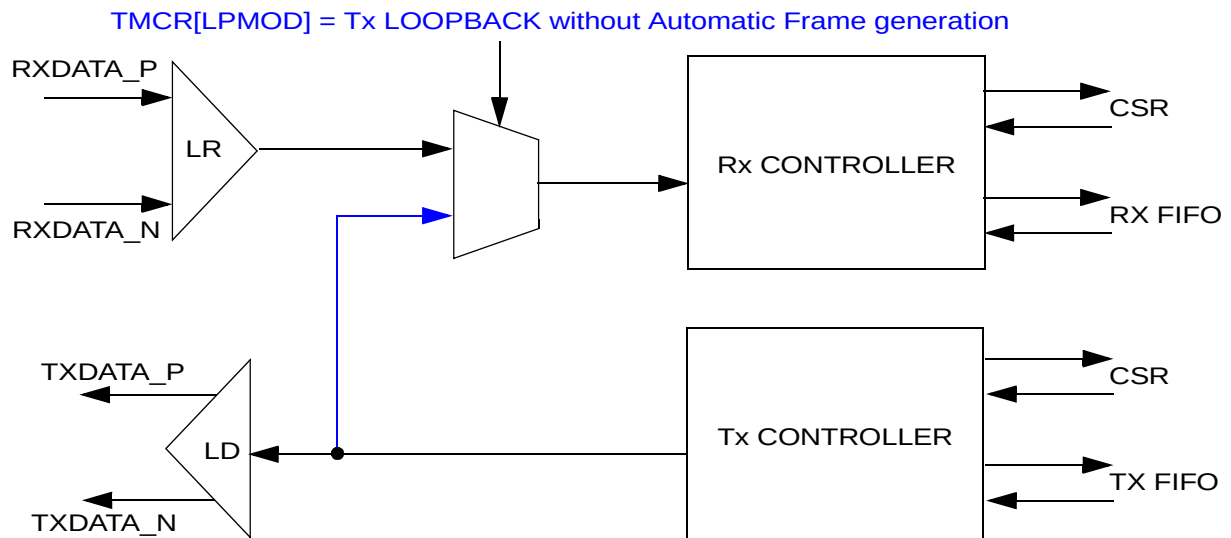
Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs `TMCR[LPON] = 0`.
- For LFAST Slave:
 - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.

49.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

Figure 840. Tx LoopBack mode without automatic frame generation



Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).

All frames and error flags are decoded in this mode.

49.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where predefined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

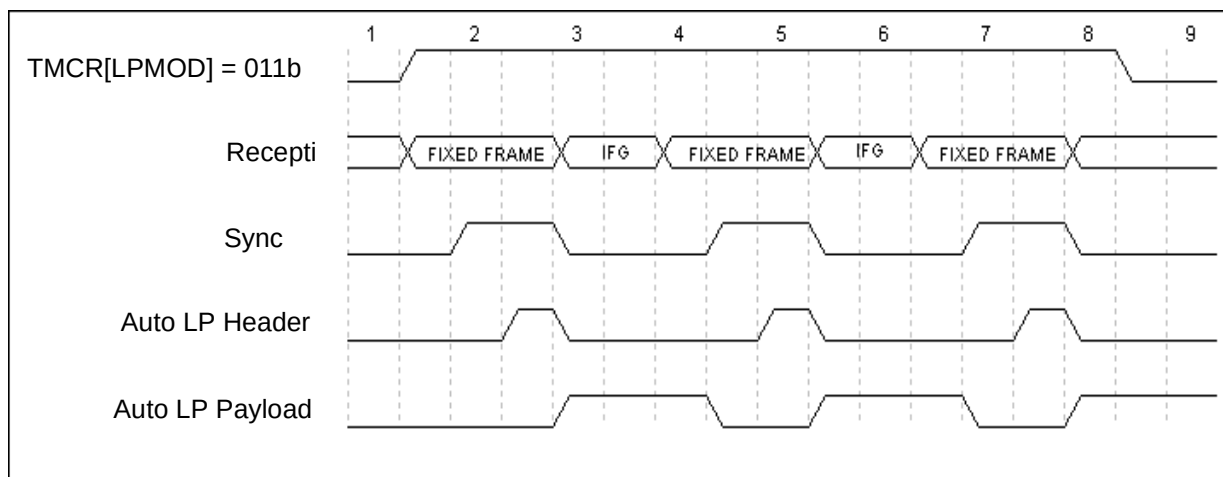
The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted.
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

Figure 841. Automatic Loopback Test status signal timings



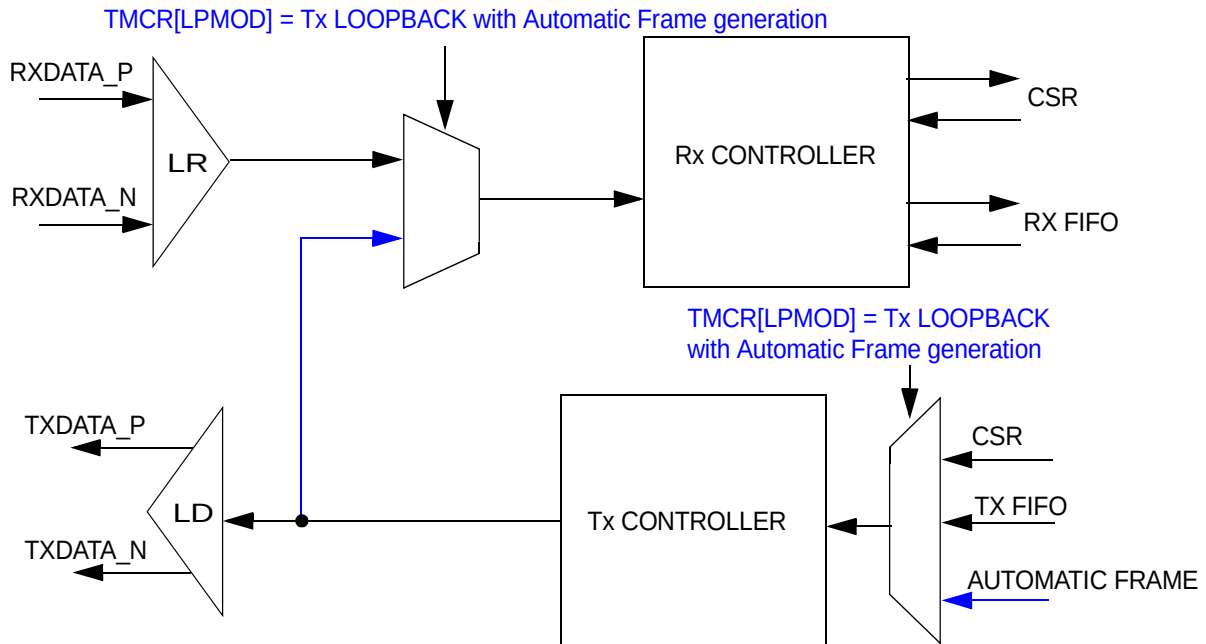
The two Loopback with automatic frame generation modes are:

- Tx Loopback with Automatic frame generation.
- Tx LVDS (external) with Automatic frame generation.

49.7.7.1.4.1 Tx LoopBack mode with automatic frame generation

When **TMCR[LPMOD] = 011b** (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

Figure 842. Tx LoopBack mode with automatic frame generation



Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from the LFAST Master.

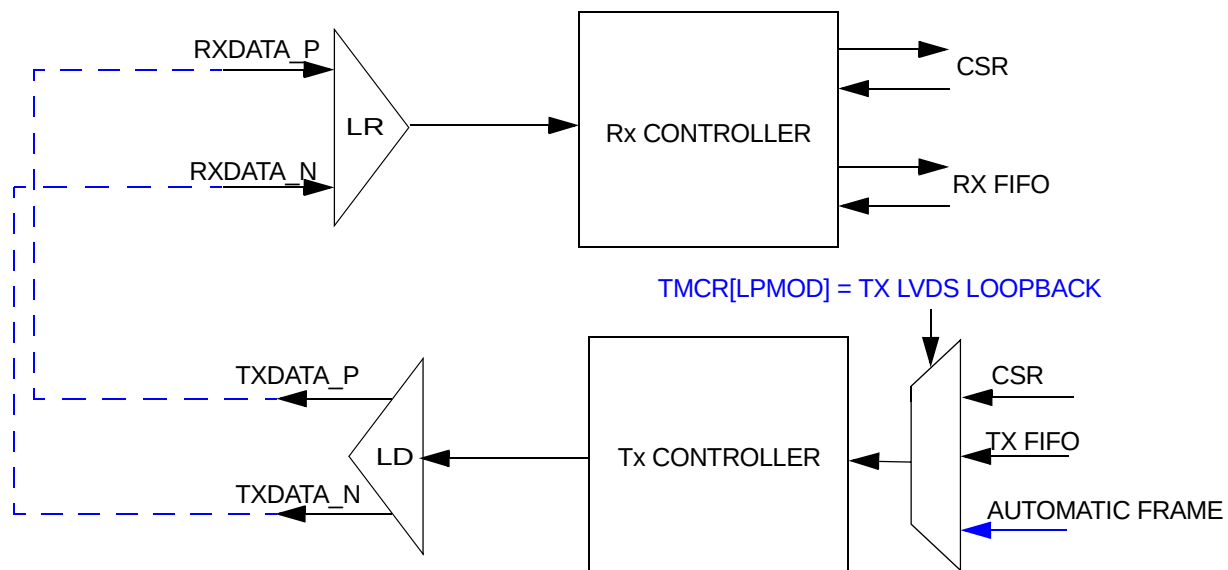
Exit from Tx Loopback with Automatic frame generation mode can be done by the following method:

- S/W programs TMCR[LPON] = 0.

49.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

Figure 843. Tx LVDS Loopback (external) mode with automatic frame generation



Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on) from LFAST Master

Exit from Tx LVDS Loopback with automatic frame generation mode can be done by the following method:

- S/W programs TMCR[LPON] = 0.

49.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 34h (Clock Test mode on) from LFAST Master.

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by any of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master.

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode does not affect the functionality of the Rx Interface.

49.7.8 Interrupts

The LFAST module generates five interrupts to the processor:

- Transmit complete interrupt
- Transmit exception interrupt
- Receive complete interrupt
- Receive exception interrupt
- ICLC receive interrupt

Each interrupt is asserted when any one of their status conditions is met.

49.7.8.1 Transmit complete interrupt

This interrupt indicates a transfer completion of a frame. This interrupt is asserted whenever any of the following bits are set in the TISR and also the corresponding mask bits are set in the TIER.

- TXDTF
- TXICLCF
- TXUNSF
- TXPNGF

Each of these bits can be individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

49.7.8.2 Transmit exception interrupt

This interrupt indicates occurrence of an exception condition in the Tx block. This interrupt is asserted whenever any of the following bits are set in the TISR along with the corresponding mask bits in the TIER.

- TXOVF
- TXIEF

Each of these bits can be individually masked. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

Table 869. Recommended transmit exception handling

Exception	Recommendation
TXOVF	– MCR[TXEN] = 0 – The system level interfaces transmit is reseted and enabled again – For Slave: After reception of ICLC Ping Response Request the MCR[TXEN] may be set – For Master: The MCR[TXEN] should be set and an ICLC frame Ping Response Request should be sent
TXIEF	– The MCR[TXEN] should be set

49.7.8.3 Receive complete interrupt

This interrupt indicates a reception of a frame. This interrupt is asserted whenever any of the following bits is set in the RISR along with the corresponding interrupt enable bit is also set in the RIER.

- RXCTSF
- RXDF
- RXUNSF

Each of these bits are individually maskable. So, the bits that are not required to generate an interrupt to the processor can be masked by clearing its bit in the RIER. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

49.7.8.4 Receive exception interrupt

This interrupt indicates occurrence of an exception condition in Rx block. This interrupt is asserted whenever any of the following bits is set in the RISR and corresponding enable mask bit is also set in the RIER.

- RXLCEF
- RXICF
- RXSZF
- RXUOF
- RXUFF
- RMXF
- RXMNF
- RXOFF

Each of these bits are individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked by clearing the appropriate bit in the RIER. For details on the cause of assertion of each bit refer to the RISR description. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

Table 870. Recommended receive exception handling mechanism

Exception	Recommendation
RXLCEF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXICF	No action required
RXSZF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXOFF	– The MCR[RXEN] may be cleared – The system level interfaces receive may be reset and enabled, as current transfer may be corrupted
RXUFF	No action required
RMXF	The system level interfaces receive should be enabled, if not enabled
RXMNF	No action required
RXUNSF	No action required

49.7.8.5 ICLC receive interrupt

This interrupt indicates reception of a valid ICLC frame. This interrupt is asserted whenever any of the following bits is set in the RIISR along with the corresponding enable bit mask bit is set in the RIIER.

- ICPONF
- ICPOFF
- ICTSF
- ICTFF
- ICRSF
- ICRFF
- ICTEF
- ICTDF
- ICCTF
- ICLPF
- ICTOF
- ICPRF
- ICPSF
- ICPFF

Each of these bits are individually maskable. So those bits that are not required to generate an interrupt to the processor can be masked. For details on the cause of assertion of each bit, refer RIISR. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in RIISR.

49.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has its own dedicated buffer, and they are not shared between each other.

Table 871. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer (in bits)	Rx Buffer (in bits)	Memory Type
Data Frame	38 × 32 max 6 packets	38 × 32 max 6 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	1 × 7 max 1 packet ⁽¹⁾	1 × 8 max 1 packet ⁽²⁾	Registers ICR, PISR
CTS Frame	N/A	N/A	N/A

1. Only for MCR[MSEN] = 1

2. Only for ping response data when MCR[MSEN] = 1

49.9 Resets

The various blocks in LFAST are reset as described in [Table 872](#).

Table 872. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ]

49.10 Clocks

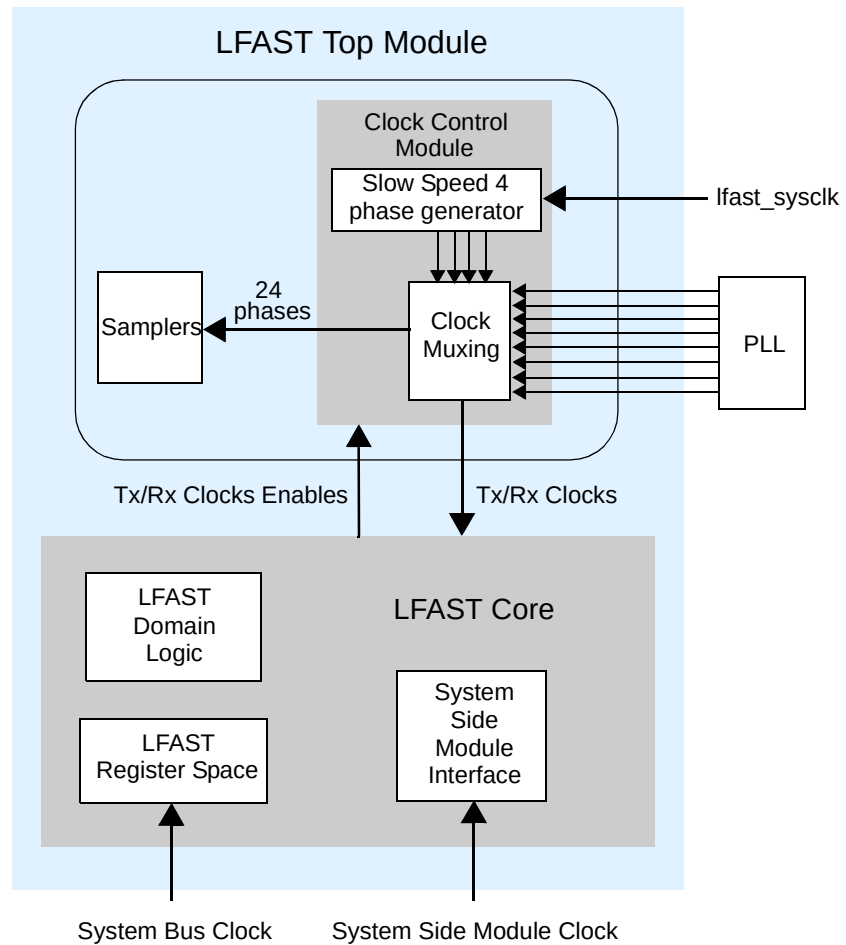
The LFAST mainly works on three clocks:

- System Bus Clock, used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

49.10.1 Clocking strategy

[Figure 844](#) shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

Figure 844. LFAST module clock domains

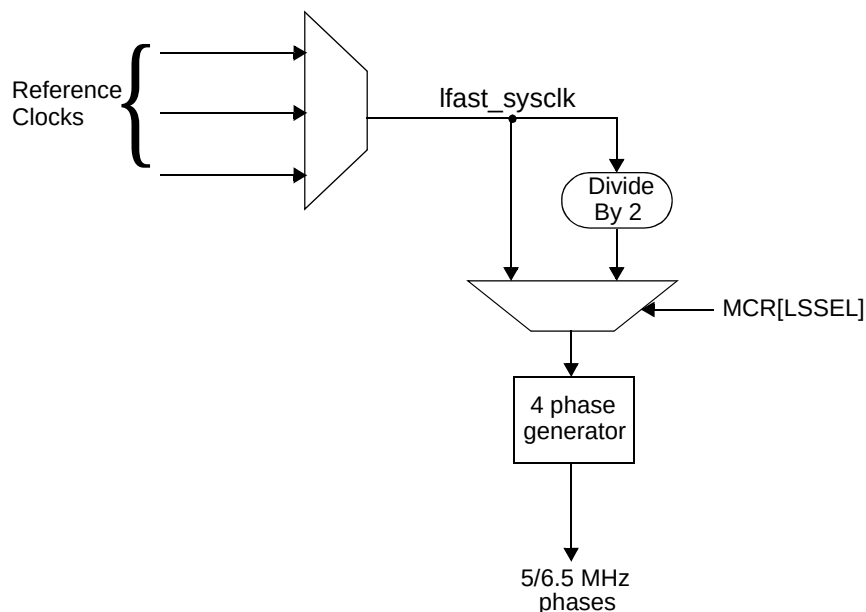


49.10.2 Slow speed clock

49.10.2.1 External muxing

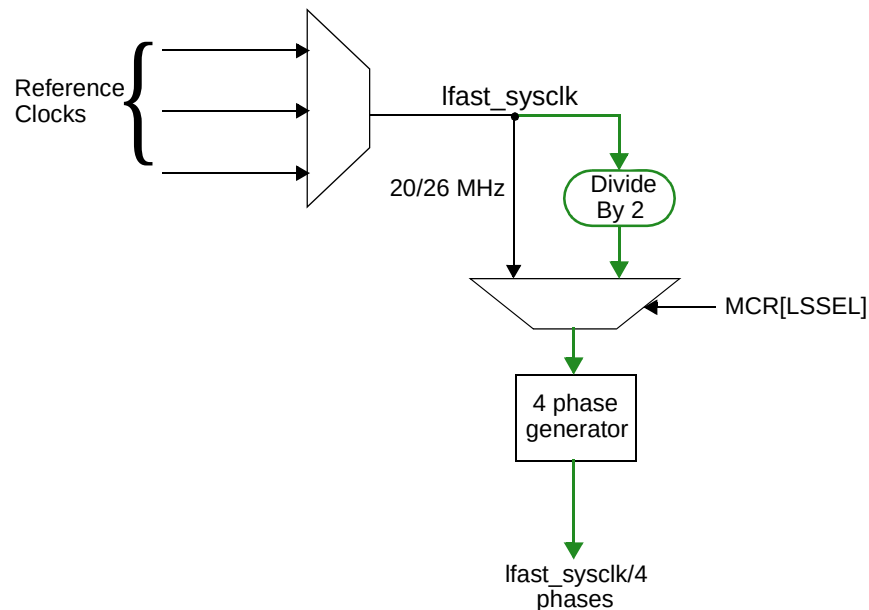
The Clock Control Module will receive `lfast_sysclk` from the clocking subsystem as shown in [Figure 845](#), [Figure 846](#) and [Figure 847](#).

Figure 845. External clock muxing of lfast_sysclk

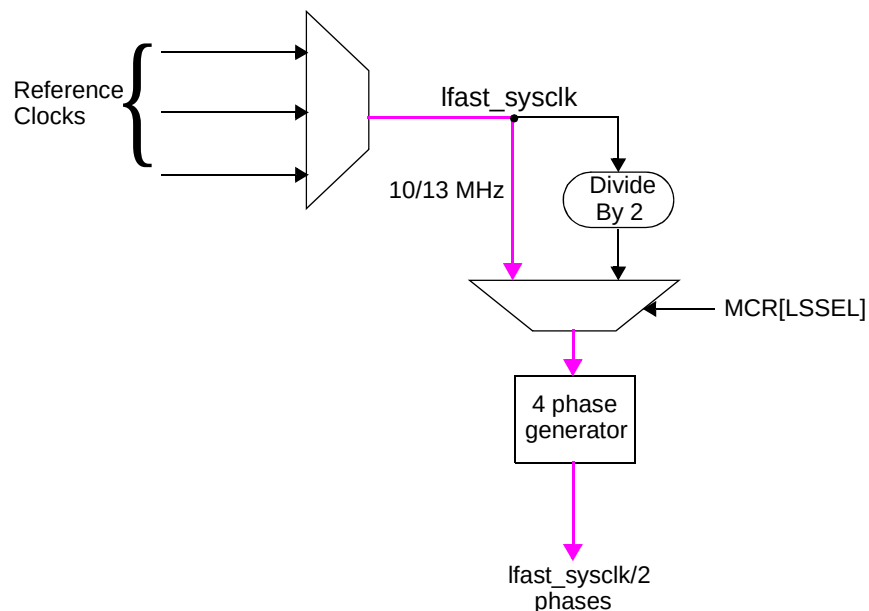


All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in [Figure 845](#).

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in [Figure 846](#) (selected clock path shown in green).

Figure 846. External clock muxing of lfast_sysclk of 20/26 MHz

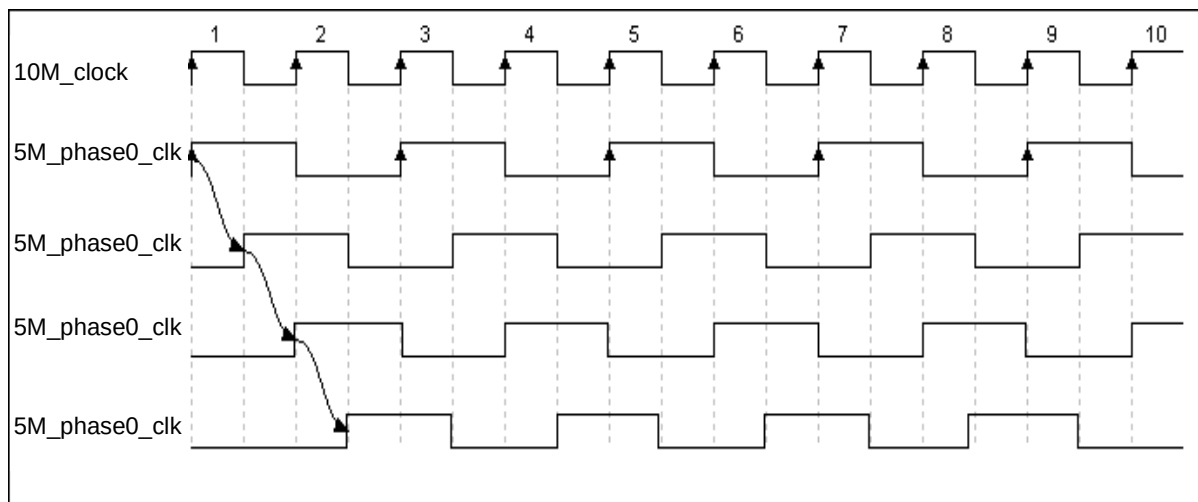
In this case, `lfast_sysclk` is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency $\text{lfast_sysclk}/2$ when `MCR[LSSEL] = 0`. If `lfast_sysclk` is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in [Figure 847](#) (selected clock path selected shown in magenta).

Figure 847. External clock muxing of lfast_sysclk of 10 MHz

49.10.2.2 Slow speed 4-phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. [Figure 848](#) Shows 4 phases getting generated using lfast_sysclk.

Figure 848. Slow speed 4 phases generation

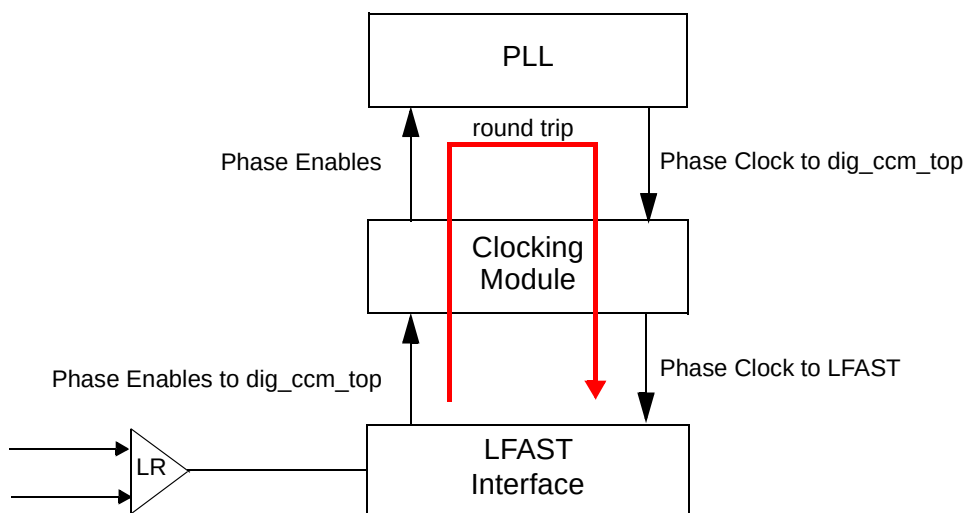


49.10.3 Rx controller clocks

49.10.3.1 Clocking module requirements for high speed phases

The high speed Phases are obtained from the PLL via the Clocking Module as shown in [Figure 849](#).

Figure 849. Clock Enables and Clock Paths



The LFAST block will know which of the phases will be required for the different modes of operations. Therefore, the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

49.10.3.2 Clock module requirements for low speed phases

49.10.3.2.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq $\times$ 8 MHz clock from the PLL is not needed.

Table 873. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

49.10.4 Tx controller clocks

49.10.4.1 Clocking module requirements for speed phases

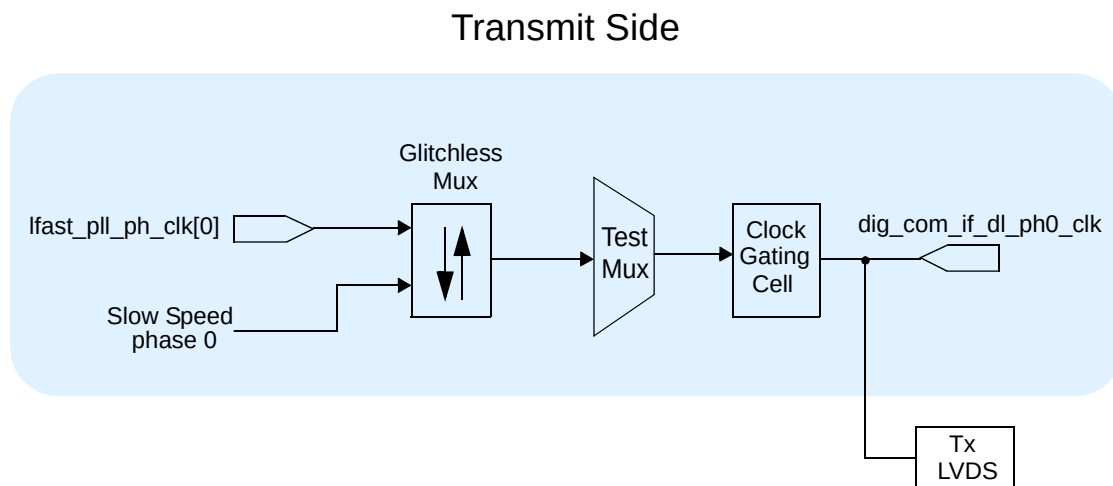
The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See [Table 874](#).

Table 874. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

49.10.4.2 Transmit clock muxing

Figure 850. Transmit Clocks Muxing



49.10.4.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

50 Fast Ethernet Controller (FEC)

50.1 Introduction

The Fast Ethernet Controller (FEC) is a communication controller that supports 10 and 100 Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports five different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII, 10/100 Mbps reduced MII, and the 10 Mbps-only 7-wire interface.

50.1.1 Block diagram

[Figure 851](#) shows the block diagram of the FEC. The FEC is implemented with a combination of hardware and microcode. The off-chip (Ethernet) interfaces are compliant with industry and IEEE 802.3 standards.

Figure 851. FEC block diagram

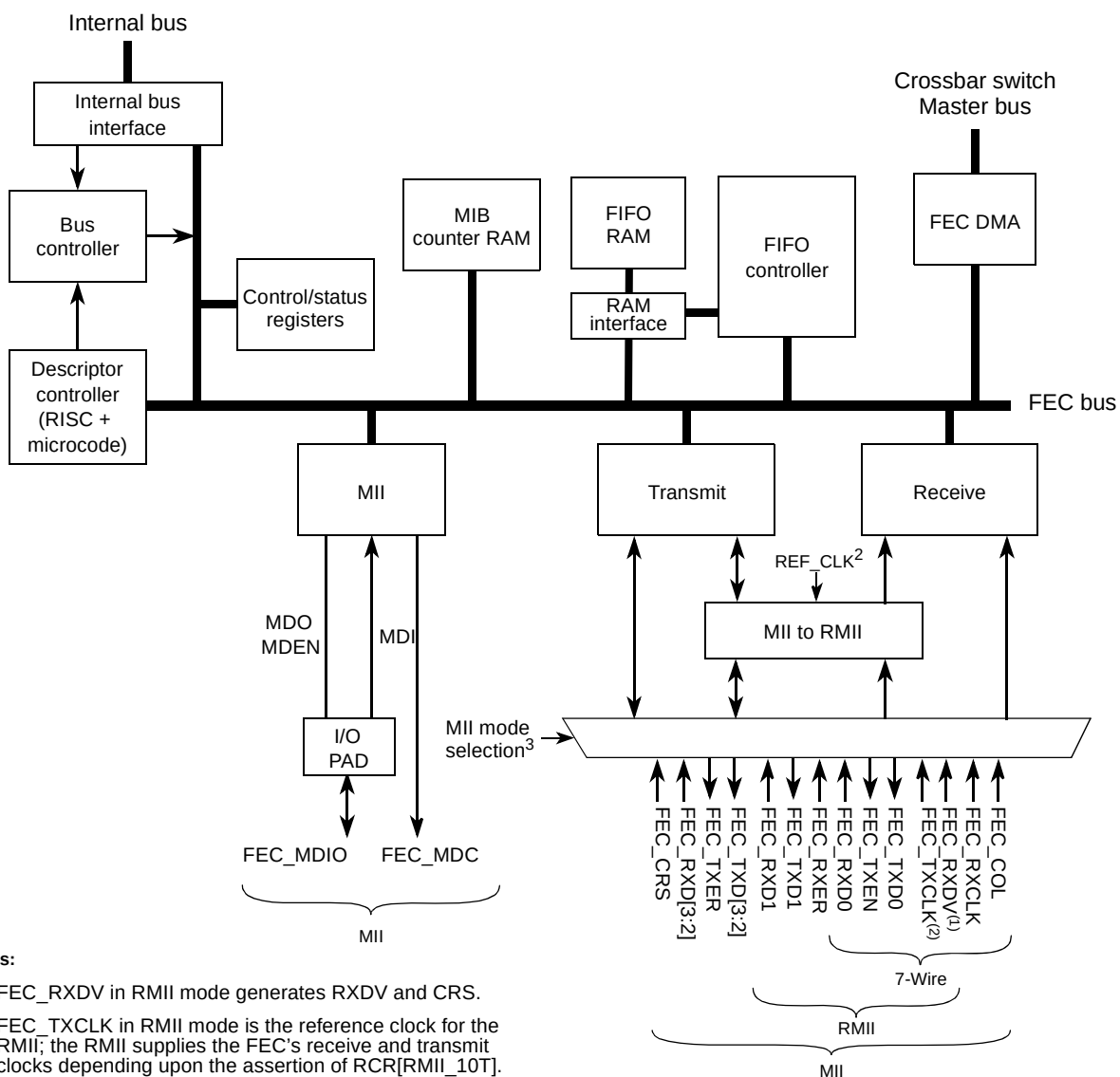


Table 875 provides a brief description of the blocks in Figure 851.

Table 875. FEC sub-block descriptions

Block	Description
Control and status registers (CSR)	Provides global control (Ethernet reset and enable) and interrupt managing registers
Descriptor controller	A RISC-based controller providing these functions in the FEC: – Initialization (those internal registers not initialized by you or hardware) – High-level control of the DMA channels (initiating DMA transfers) – Interpreting buffer descriptors – Address recognition for receive frames – Random number generation for transmit collision backoff timer
DMA block	Provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to all run independently <i>Note: In this document, "DMA" refers to the DMA block within the FEC, and is not related to any other DMA controllers that may be present on the chip.</i>
Media independent interface (MII)	Provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of: – The management data clock (FEC_MDC) – The management data input/output lines (FEC_MDIO) of the MII interface
Message information block (MIB)	Maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC, but provides valuable counters for network management. The counters supported are: – The RMON (RFC 1757) Ethernet Statistics group – Some of the IEEE 802.3 counters See the memory map for more information.
RAM	Serves as the focal point of all data flow in the Fast Ethernet controller and divides into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to/from the DMA block from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block. Receive data flows from the receive block into the receive FIFO.
Transmit and receive blocks	Provide the Ethernet MAC functionality (with some assist from microcode)

50.1.2 Features

The FEC incorporates the following features:

- Support for the following Ethernet physical interfaces:^(o)
 - 100-Mbps IEEE 802.3 MII
 - 10-Mbps IEEE 802.3 MII
 - 100-Mbps reduced media independent interface (RMII)
 - 10-Mbps reduced media independent interface (RMII)
 - 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200 Mbps throughput) with a minimum internal bus clock rate of 50 MHz
- Support for half-duplex operation (100 Mbps throughput) with a minimum internal bus clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
 - Frames with broadcast address may be always accepted or always rejected
 - Exact match for single 48-bit individual (unicast) address
 - Hash (64-bit hash) check of individual (unicast) addresses
 - Hash (64-bit hash) check of group (multicast) addresses
 - Promiscuous mode

50.2 Modes of operation

This section describes the FEC operating modes.

50.2.1 Full and half duplex operation

Full duplex mode is for use on point-to-point links between switches or end node to switch. Half duplex mode works in connections between an end node and a repeater or between repeaters. TCR[FDEN] controls duplex mode selection.

When configured for full duplex mode, flow control may be enabled. For more details, see:

- The RFC_PAUSE and TFC_PAUSE fields in the TCR ([Section 50.4.10, Transmit Control Register \(TCR\)](#))
- The RCR[FCE] field ([Section 50.4.9, Receive Control Register \(RCR\)](#))
- [Section 50.5.13, Full duplex flow control](#)

^o. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

50.2.2 Interface options^(p)

The following interface options are supported. A detailed discussion of the interface configurations is provided in [Section 50.5.8, Network interface options](#).

50.2.2.1 10 Mbps and 100 Mbps MII interface

The IEEE 802.3 standard defines the media independent interface (MII) for 10/100 Mbps operation. The MAC-PHY interface may be configured to operate in MII mode by programming RCR[MII_MODE] = 1.

The operation speed is determined by the FEC_TXCLK and FEC_RXCLK pins driven by the external transceiver. The transceiver auto-negotiates the speed or software controls it via the serial management interface (FEC_MDC/FEC_MDIO pins) to the transceiver. Refer to the MMFR and MSCR register descriptions, as well as the section on the MII, for a description of how to read and write registers in the transceiver via this interface.

50.2.2.2 10 Mbps and 100 Mbps RMII interface

The reduced media independent interface (RMII) is a low-cost alternative to the IEEE 802.3 MII standard. This interface provides the functionality of the MII interface on a total of 8 pins instead of 18. The RMII interface for 10/100 Ethernet MAC-PHY interface was defined by an industry consortium and is not currently included in the IEEE 802.3 standard. This functionality is selected as described in the chip configuration chapter, and is reflected in the read-only RCR[RMII_MODE] bit. The RCR[RMII_10T] bit determines the speed of operation. The reference clock for RMII is always 50 MHz, but this clock can be divided by 10 within the RCR register to support 10 Mbps operation. The PHY must be configured accordingly.

50.2.2.3 10 Mbps 7-wire interface operation

The FEC supports 7-wire interface used by many 10 Mbps Ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is cleared, MII mode is disabled and the 10 Mbps 7-wire mode is enabled.

50.2.3 Address recognition options

The address options supported are:

- Promiscuous
- Broadcast reject
- Individual address (hash or exact match)
- Multicast hash match

Address recognition options are discussed in detail in [Section 50.5.11, Ethernet address recognition](#).

50.2.4 Internal loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in [Section 50.5.16, MII internal and external loopback](#).

p. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

50.3 External signal description

[Table 876](#) describes the various FEC signals, as well as indicating which signals work in available modes.

Table 876. FEC signal descriptions⁽¹⁾

Signal name	MII	7-wire	RMII	Description
FEC_COL	X	X	—	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.
FEC_CRD	X	—	—	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the FEC_RXDV pin.
FEC_MDC	X	—	—	Output clock provides a timing reference to the PHY for data transfers on the FEC_MDIO signal.
FEC_MDIO	X	—	—	Transfers control information between the external PHY and the media-access controller. Data is synchronous to FEC_MDC. This signal is an input after reset. When the FEC operates in 10Mbps 7-wire interface mode, this signal should be connected to VSS.
FEC_RXCLK	X	X	—	Provides a timing reference for FEC_RXDV, FEC_RXD[3:0], and FEC_RXER.
FEC_RXDV	X	X	X	Asserting the FEC_RXDV input indicates PHY has valid nibbles present on the MII. FEC_RXDV must remain asserted from the first recovered nibble of the frame through to the last nibble. Assertion of FEC_RXDV must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal.
FEC_RXD0	X	X	X	This pin contains the Ethernet input data transferred from PHY to the media-access controller when FEC_RXDV is asserted.
FEC_RXD1	X	—	X	This pin contains the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXD[3:2]	X	—	—	These pins contain the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXER	X	—	X	When asserted with FEC_RXDV, indicates PHY detects an error in the current frame. When FEC_RXDV is not asserted, FEC_RXER has no effect.
FEC_TXCLK	X	X	X	Input clock which provides a timing reference for FEC_TXEN, FEC_TXD[3:0] and FEC_TXER. In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.
FEC_TXD0	X	X	X	The serial output Ethernet data and only valid during the assertion of FEC_TXEN.
FEC_TXD1	X	—	X	This pin contains the serial output Ethernet data and valid only during assertion of FEC_TXEN.
FEC_TXD[3:2]	X	—	—	These pins contain the serial output Ethernet data and valid only during assertion of FEC_TXEN.

Table 876. FEC signal descriptions⁽¹⁾ (continued)

Signal name	MII	7-wire	RMII	Description
FEC_TXEN	X	X	X	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is negated before the first FEC_TXCLK following the final nibble of the frame.
FEC_TXER	X	—	—	When asserted for one or more clock cycles while FEC_TXEN is also asserted, PHY sends one or more illegal symbols. FEC_TXER has no effect at 10 Mbps or when FEC_TXEN is negated.

1. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

50.4 Memory map and register definition

The FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs control operation modes and extract global status information. The descriptors pass data buffers and related buffer information between the hardware and software.

Each FEC implementation requires a 1 KB memory map space, which is divided into two sections of 512 bytes each for:

- Control/status registers
- Event/statistic counters held in the MIB block

[Table 877](#) defines the top-level memory map.

Table 877. FEC memory map

Address offset	Function
000h–1FFh	Control/status registers
200h–3FFh	MIB block counters

[Table 878](#) shows the FEC register map. Any address offset not explicitly mentioned in [Table 878](#) is reserved. Do not modify the content of reserved fields. All accesses to and from the registers must be via 32-bit accesses. There is no support for accesses other than 32-bit. Accessing reserved or unimplemented register spaces in the FEC will not return transfer errors.

The MIB counters (shown in [Table 878](#)) define the locations in the MIB RAM space where hardware-maintained counters reside. The counters are divided into two groups:

- RMON counters include the Ethernet statistics counters defined in RFC 1757
- Truncated frames counter (only frames with lengths up to 2047 bytes are supported)

The transmit and receive RMON counters are independent, which ensures accurate network statistics when operating in full duplex mode.

The included IEEE counters support the mandatory and recommended counter packages defined in Section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The FEC supports IEEE Basic Package objects, but these do not require counters in the MIB block. In addition, some of the recommended package objects supported do not require MIB counters. Counters for transmit and receive full duplex flow control frames are also included.

Table 878. FEC register map

Address offset	Register	Location
004h	Interrupt Event Register (EIR)	on page 1822
008h	Interrupt Mask Register (EIMR)	on page 1824
010h	Receive Descriptor Active Register (RDAR)	on page 1825
014h	Transmit Descriptor Active Register (TDAR)	on page 1826
024h	Ethernet Control Register (ECR)	on page 1827
040h	MII Management Frame Register (MMFR)	on page 1827
044h	MII Speed Control Register (MSCR)	on page 1829
064h	MIB Control Register (MIBC)	on page 1830
084h	Receive Control Register (RCR)	on page 1831
0C4h	Transmit Control Register (TCR)	on page 1832
0E4h	Physical Address Low Register (PALR)	on page 1833
0E8h	Physical Address High Register and Type Field (PAUR)	on page 1834
0ECh	Opcode/Pause Duration (OPD)	on page 1834
118h	Descriptor Individual Upper Address Register (IAUR)	on page 1835
11Ch	Descriptor Individual Lower Address Register (IALR)	on page 1835
120h	Descriptor Group Upper Address Register (GAUR)	on page 1836
124h	Descriptor Group Lower Address Register (GALR)	on page 1836
144h	Transmit FIFO Watermark (TFWR)	on page 1837
14Ch	FIFO Receive Bound Register (FRBR)	on page 1837
150h	FIFO Receive Start Register (FRSR)	on page 1838
180h	Receive Descriptor Ring Start Register (ERDSR)	on page 1838
184h	Transmit Buffer Descriptor Ring Start Register (ETDSR)	on page 1839
188h	Receive Buffer Size Register (EMRBR)	on page 1839
200h	Count of frames not counted correctly (RMON_T_DROP)	on page 1840
204h	RMON Tx packet count (RMON_T_PACKETS)	on page 1840
208h	RMON Tx broadcast packets (RMON_T_BC_PKT)	on page 1841
20Ch	RMON Tx multicast packets (RMON_T_MC_PKT)	on page 1841
210h	RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN)	on page 1842
214h	RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)	on page 1842
218h	RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)	on page 1842

Table 878. FEC register map (continued)

Address offset	Register	Location
21Ch	RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)	on page 1843
220h	RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB)	on page 1843
224h	RMON Tx collision count (RMON_T_COL)	on page 1843
228h	RMON Tx 64 byte packets (RMON_T_P64)	on page 1844
22Ch	RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)	on page 1844
230h	RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)	on page 1844
234h	RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)	on page 1845
238h	RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)	on page 1845
23Ch	RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047)	on page 1845
240h	RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)	on page 1846
244h	RMON Tx Octets (RMON_T_OCTETS)	on page 1846
248h	Count of transmitted frames not counted correctly (IEEE_T_DROP)	on page 1846
24Ch	Frames transmitted OK (IEEE_T_FRAME_OK)	on page 1847
250h	Frames transmitted with single collision (IEEE_T_1COL)	on page 1847
254h	Frames transmitted with multiple collisions (IEEE_T_MCOL)	on page 1847
258h	Frames transmitted after deferral delay (IEEE_T_DEF)	on page 1848
25Ch	Frames transmitted with late collision (IEEE_T_LCOL)	on page 1848
260h	Frames transmitted with excessive collisions (IEEE_T_EXCOL)	on page 1848
264h	Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR)	on page 1849
268h	Frames transmitted with carrier sense error (IEEE_T_CSERR)	on page 1849
26Ch	Frames transmitted with SQE error (IEEE_T_SQE)	on page 1849
270h	Flow control pause frames transmitted (IEEE_T_FDXFC)	on page 1850
274h	Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)	on page 1850
280h	Count of received frames not counted correctly (RMON_R_DROP)	on page 1850
284h	RMON Rx packet count (RMON_R_PACKETS)	on page 1851
288h	RMON Rx broadcast packets (RMON_R_BC_PKT)	on page 1851
28Ch	RMON Rx multicast packets (RMON_R_MC_PKT)	on page 1852
290h	RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN)	on page 1852
294h	RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)	on page 1852
298h	RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)	on page 1853
29Ch	RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)	on page 1853
2A0h	RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB)	on page 1853
2A4h	Reserved (RMON_R_RESVD_0)	on page 1854
2A8h	RMON Rx 64 byte packets (RMON_R_P64)	on page 1854
2ACh	RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)	on page 1854

Table 878. FEC register map (continued)

Address offset	Register	Location
2B0h	RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)	on page 1855
2B4h	RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)	on page 1855
2B8h	RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023)	on page 1855
2BCh	RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047)	on page 1856
2C0h	RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048)	on page 1856
2C4h	RMON Rx octets (RMON_R_OCTETS)	on page 1856
2C8h	Count of received frames not counted correctly (IEEE_R_DROP)	on page 1857
2CCh	Frames received OK (IEEE_R_FRAME_OK)	on page 1857
2D0h	Frames received with CRC error (IEEE_R_CRC)	on page 1857
2D4h	Frames received with alignment error (IEEE_R_ALIGN)	on page 1858
2D8h	Receive FIFO overflow count (IEEE_R_MACERR)	on page 1858
2DCh	Flow control pause frames received (IEEE_R_FDXFC)	on page 1858
2E0h	Octet count for frames received without error (IEEE_R_OCTETS_OK)	on page 1859

50.4.1 Interrupt Event Register (EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts which may occur in normal operation are GRA, TXF, TXB, RXF, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LC, and RL. Interrupts resulting from internal errors are HBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters:

- HBERR - IEEE_T_SQE
- BABR - RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT - RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
- LATE_COL - IEEE_T_LCOL
- COL_RETRY_LIM - IEEE_T_EXCOL
- XFIFO_UN - IEEE_T_MACERR

Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

Figure 852. Interrupt Event Register (EIR)

Offset: 004h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0	0	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 879. EIR field descriptions

Field	Description
HBERR	Heartbeat error. Indicates TCR[HBC] is set and that the COL input was not asserted within the heartbeat window following a transmission.
BABR	Babbling receive error. Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
BABT	Babbling transmit error. Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused by a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
GRA	Graceful stop complete. Indicates the graceful stop is complete. During graceful stop the transmitter is placed into a pause state after completion of the frame currently being transmitted. This bit is set by one of these conditions: – A graceful stop initiated by the setting of the TCR[GTS] bit is now complete. – A graceful stop initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. – A graceful stop initiated by the reception of a valid full duplex flow control pause frame is now complete. See Section 50.5.13, Full duplex flow control.
TXF	Transmit frame interrupt. Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
TXB	Transmit buffer interrupt. Indicates a transmit buffer descriptor has been updated.
RXF	Receive frame interrupt. Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
RXB	Receive buffer interrupt. Indicates a receive buffer descriptor not the last in the frame has been updated.
MII	MII interrupt. Indicates the MII has completed the data transfer requested.
EBERR	Ethernet bus error. Indicates a system bus error occurred when a DMA transaction is underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs, software needs to ensure that the FIFO controller and DMA also soft reset.
LC	Late collision. Indicates a collision occurred beyond the collision window (slot time) in half duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.

Table 879. EIR field descriptions (continued)

Field	Description
RL	Collision retry limit. Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half duplex mode.
UN	Transmit FIFO underrun. Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.

50.4.2 Interrupt Mask Register (EIMR)

The EIMR register controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit.

Each bit in this register corresponds to an interrupt source defined by the EIR register. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is set.

Figure 853. Ethernet Interrupt Mask Register (EIMR)

Offset: 008h													Access: User read/write			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 880. EIMR field descriptions

Field	Description
HBERR	Heartbeat error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
BABR	Babbling receive error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.

Table 880. EIMR field descriptions (continued)

Field	Description
BABT	Babbling transmit error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
GRA	Graceful stop complete interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
TXF	Transmit frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
TXB	Transmit buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
RXF	Receive frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
RXB	Receive buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
MII	MII interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
EBERR	Ethernet bus error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
LC	Late collision interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
RL	Collision retry limit interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
UN	Transmit FIFO underrun interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.

50.4.3 Receive Descriptor Active Register (RDAR)

RDAR is a command register, written by the user, indicating the receive descriptor ring is updated (the driver produced empty receive buffers with the empty bit set).

When the register is written, the RDAR bit is set. This is independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (if ECR[ETHER_EN] is also set). After the FEC polls a receive descriptor whose empty bit is not set, FEC clears the RDAR bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors are placed into the receive descriptor ring.

The RDAR register is cleared at reset and when ECR[ETHER_EN] is cleared.

Figure 854. Receive Descriptor Active Register (RDAR)

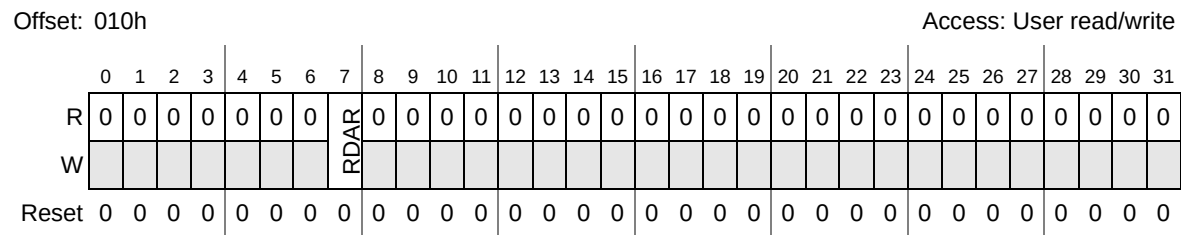


Table 881. RDAR field descriptions

Field	Description
RDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC when no additional empty descriptors remain in the receive ring. Also cleared when ECR[ETHER_EN] is cleared.

50.4.4 Transmit Descriptor Active Register (TDAR)

The TDAR is a command register which the user writes to indicate the transmit descriptor ring is updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

When the register is written, the TDAR field is set. This value is independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and processes transmit frames (provided ECR[ETHER_EN] is also set). After the FEC polls a transmit descriptor that is a ready bit not set, FEC clears the TDAR field and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors are placed into the transmit descriptor ring.

The TDAR field is cleared when any of the following events occur:

- Chip reset
- When ECR[ETHER_EN] is cleared
- When ECR[RESET] is set

Figure 855. Transmit Descriptor Active Register (TDAR)

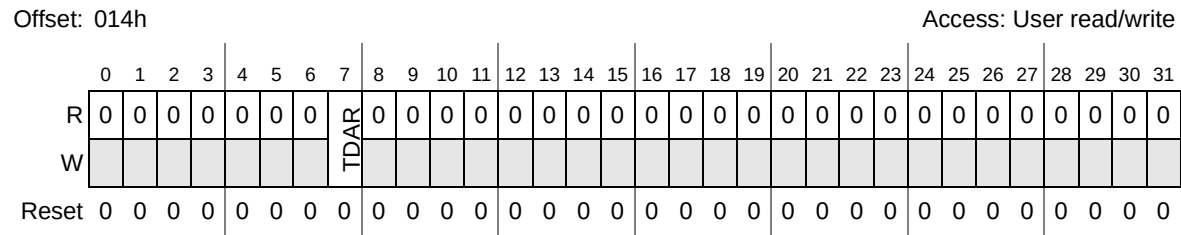


Table 882. TDAR field descriptions

Field	Description
TDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHER_EN] is cleared.

50.4.5 Ethernet Control Register (ECR)

ECR is a read/write user register, though hardware may alter fields in this register as well. The ECR enables/disables the FEC.

Figure 856. Ethernet Control Register (ECR)

Offset: 024h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ETHER_EN	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 883. ECR field descriptions

Field	Description
ETHER_EN	When this bit is set, FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware alters the ETHER_EN bit under the following conditions: – ECR[RESET] is set by software, in which case ETHER_EN is cleared. – An error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN is cleared.
RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ECR[ETHER_EN] is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately eight internal bus clock cycles after this bit is set.

50.4.6 MII Management Frame Register (MMFR)

The MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to the MMFR causes a

management frame to be sourced unless the MSCR is programmed to 0. If MSCR is cleared while MMFR is written and then MSCR is written with a non-zero value, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

The MMFR does not reset to any predefined value.

Figure 857. MII Management Frame Register (MMFR)

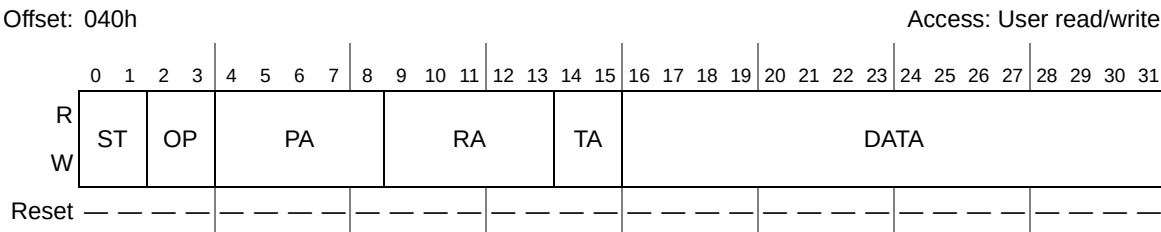


Table 884. MMFR field descriptions

Field	Description
ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.
OP	Operation code. 00 Write frame operation, but not MII compliant. 01 Write frame operation for a valid MII management frame. 10 Read frame operation for a valid MII management frame. 11 Read frame operation, but not MII compliant.
PA	PHY address. This field specifies one of up to 32 attached PHY devices.
RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

To perform a read or write operation on the MII Management Interface, write the MMFR. To generate a valid read or write management frame, ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated, but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII Management Interface write frame (write to a PHY register), write {01 01 PHYAD REGAD 10 DATA} to the MMFR. Writing this pattern causes the control logic to shift out the data in the MMFR register following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the write management frame operation completes, the MII interrupt is generated. At this time, contents of the MMFR match the original value written.

To generate an MII management interface read frame (read a PHY register), write {01 10 PHYAD REGAD 10 XXXX} to the MMFR (the content of the DATA field is unimportant). Writing this pattern causes the control logic to shift out the data in the MMFR register following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the read management frame operation completes, the MII interrupt is generated. At this time, the contents of the MMFR register match the original value written.

except for the DATA field whose contents are replaced by the value read from the PHY register.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the MII interrupt to avoid writing to the MMFR while frame generation is in progress.

50.4.7 MII Speed Control Register (MSCR)

The MSCR:

- Provides control of the MII clock (FEC_MDC pin) frequency
- Allows a preamble drop on the MII management frame

Figure 858. MII Speed Control Register (MSCR)

Offset: 044h																				Access: User read/write												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIS PRE	MII_SPEED						0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 885. MSCR field descriptions

Field	Description
DIS_PRE	Setting this bit causes the preamble (32 ones) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
MII_SPEED	Controls the frequency of the MII management interface clock (FEC_MDC) relative to the internal bus clock. A value of 0 in this field turns off the FEC_MDC and leaves it in low voltage state. Any non-zero value results in the FEC_MDC frequency of $1/(MII_SPEED \times 2)$ of the internal bus frequency.

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be set to 0 to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED changes during operation (change takes effect following a rising or falling edge of FEC_MDC).

If the internal bus clock is 25 MHz, programming this register to 0000_0005h results in an FEC_MDC as stated the equation below.

Equation 37 $25 \text{ MHz} \times \frac{1}{5 \times 2} = 2.5 \text{ MHz}$

[Table 886](#) shows optimum values for MII_SPEED as a function of internal bus clock frequency.

Table 886. Programming examples for MSCR

Internal FEC clock frequency	MSCR[MII_SPEED]	FEC_MDC frequency
25 MHz	5h	2.50 MHz
33 MHz	7h	2.36 MHz
40 MHz	8h	2.50 MHz
50 MHz	Ah	2.50 MHz
66 MHz	Eh	2.36 MHz
80 MHz	10h	2.50 MHz
83 MHz	10h	2.59 MHz

50.4.8 MIB Control Register (MIBC)

The MIBC is a read/write register controlling and observing the state of the MIB block. User software accesses this register if there is a need to disable the MIB block operation. For example, to clear all MIB counters in RAM:

1. Disable the MIB block
2. Clear all the MIB RAM locations
3. Enable the MIB block

The MIB_DIS bit is reset to 1. See the memory map for the locations of the MIB counters.

Figure 859. MIB Control Register (MIBC)

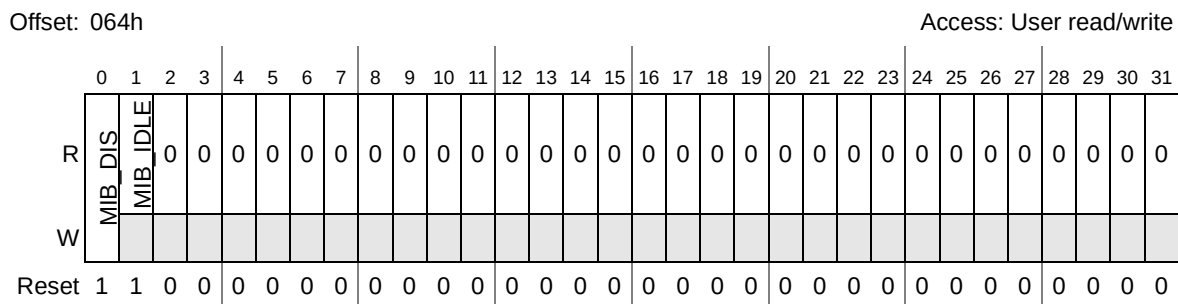


Table 887. MIBC field descriptions

Field	Description
MIB_DIS	If MIB_DIS = 1, the MIB logic halts and not update any MIB counters.
MIB_IDLE	If MIB_IDLE = 1, the MIB block is not currently updating any MIB counters.

50.4.9 Receive Control Register (RCR)

RCR controls the operational mode of the receive block and must be written only when ECR[ETHER_EN] is cleared (initialization time).

Figure 860. Receive Control Register (RCR)

Offset: 084h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	MAX_FL										
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	0	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	RMII_ECHO	RMII_LOOP	RMII_10T	RMII_MODE ⁽¹⁾	0	0	FCE	BC_REJ	PROM	MII_MODE ⁽¹⁾	DRT	LOOP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

Table 888. RCR field descriptions

Field	Description
MAX_FL	Maximum frame length. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
RMII_ECHO	RMII Echo. Enables RMII echo mode. RMII 2-bit receive data is processed through the RMII receive logic to the FEC and also routes through the RMII's transmit logic to become RMII 2-bit transmit data out. 0 Normal operation. 1 RMII echo mode enabled. <i>Note: When RMII_ECHO is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_LOOP = 0 and LOOP = 0.</i>
RMII_LOOP	RMII loopback. Enables RMII loopback mode. Causes the MII transmit outputs from the Ethernet controller to loop back to the Ethernet controllers's MII receive inputs through the RMII transmit/receive logic. 0 Normal operation. 1 RMII loopback mode enabled. <i>Note: When RMII_LOOP is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_ECHO = 0, LOOP = 0, and TCRn[FDEN] = 1.</i>

Table 888. RCR field descriptions (continued)

Field	Description
RMII_10T	RMII 10-Base T. Enables 10Mbps mode of the RMII. Determines the clock frequency of the clock source to the FEC logic to support 10/100Mbps operations. 0 100 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is sent to the RMII, while a divided-by-2 version (25 MHz) is sent to the FEC. 1 10 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is divided by 10 (5 MHz) and sent to the RMII, while a divided-by-20 version (2.5 MHz) is sent to the FEC.
RMII_MODE	RMII Mode. Indicates if the FEC is in RMII or MII/7-wire mode. See the chip configuration chapter for more details. 0 FEC is in true MII mode. 1 FEC is in RMII mode. <i>Note: Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.</i>
FCE	Flow control enable. If asserted, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) equal to FFFF_FFFF_FFFF are rejected unless the PROM bit is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the M (MISS) is set in the receive buffer descriptor.
PROM	Promiscuous mode. All frames are accepted regardless of address matching.
MII_MODE	Media independent interface mode. Selects the external interface mode for transmit and receive blocks. 0 7-wire mode (used only for serial 10 Mbps) 1 MII or RMII mode as indicated by the RMII_MODE bit <i>Note: Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.</i>
DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).
LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and transmit output signals are not asserted. The internal bus clock substitutes for the FEC_TXCLK when LOOP is asserted. DRT must be set to 0 when setting LOOP.

50.4.10 Transmit Control Register (TCR)

The TCR configures the transmit block.

Figure 861. Transmit Control Register (TCR)

Offset: 0C4h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RFC_PAUSE	TFC_PAUSE	FDEN	HBC	GTS
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

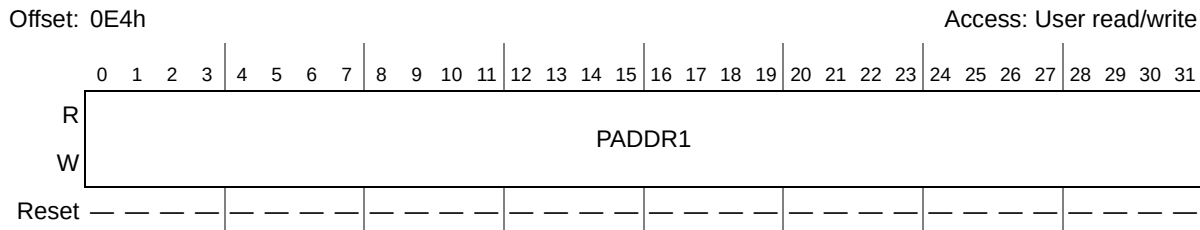
Table 889. TCR field descriptions

Field	Description
RFC_PAUSE	Receive frame control pause. This read-only status bit is asserted when a full duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, GRA interrupt in the EIR is asserted. With transmission of data frames stopped, MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC Control PAUSE frame.
FDEN	Full duplex enable. If set, frames transmit independent of carrier sense and collision inputs. This field must be modified only when ECR[ETHER_EN] = 0.
HBC	Heartbeat control. If set, the heartbeat check performs following end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This field must be modified only when ECR[ETHER_EN] = 0.
GTS	Graceful transmit stop. When this bit is set, MAC stops transmission after any frame currently transmitted is complete and GRA interrupt in the EIR is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this situation, program ECR[ETHER_EN] = 0 following the GRA interrupt.

50.4.11 Physical Address Lower Register (PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the 6-byte source address field when transmitting PAUSE frames.

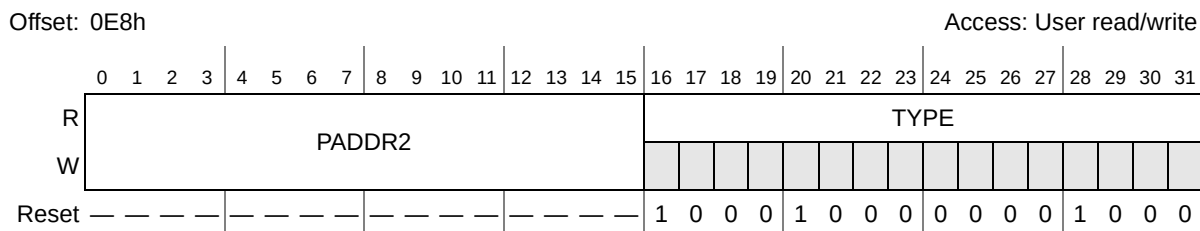
This register is not reset and you must initialize it.

Figure 862. Physical Address Lower Register (PALR)

Table 890. PALR field descriptions

Field	Description
PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

50.4.12 Physical Address Upper Register (PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte Source Address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (8808h) for transmission of PAUSE frames. The upper 16 bits of this register are not reset and you must initialize it.

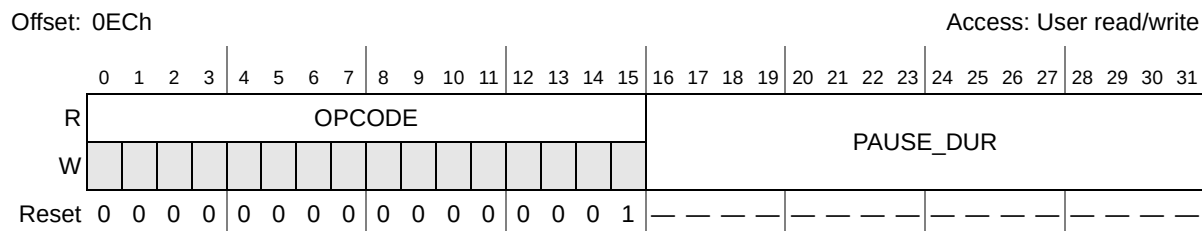
Figure 863. Physical Address Upper Register (PAUR)

Table 891. PAUR field descriptions

Field	Description
PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
TYPE	Type field in PAUSE frames. These 16 read-only bits are a constant value of 8808h.

50.4.13 Opcode/Pause Duration Register (OPD)

The OPD contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0001h. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field.

The lower 16 bits of this register are not reset and you must initialize them.

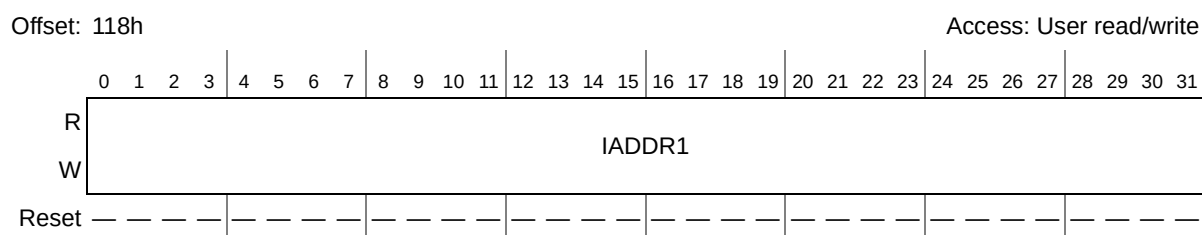
Figure 864. Opcode/Pause Duration Register (OPD)**Table 892. OPD field descriptions**

Field	Description
OPCODE	Opcode field used in PAUSE frames. These read-only bits are a constant, 0001h.
PAUSE_DUR	Pause Duration field used in PAUSE frames.

50.4.14 Descriptor Individual Upper Address Register (IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA.

This register is not reset and you must initialize it.

Figure 865. Descriptor Individual Upper Address Register (IAUR)**Table 893. IAUR field descriptions**

Field	Description
IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

50.4.15 Descriptor Individual Lower Address Register (IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA.

This register is not reset and you must initialize it.

Figure 866. Descriptor Individual Lower Address Register (IALR)

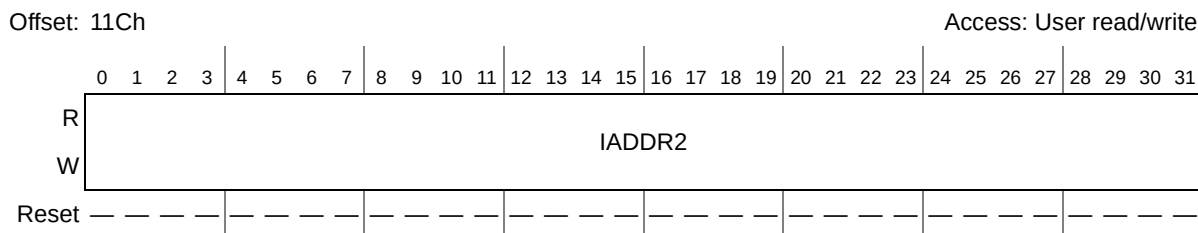


Table 894. IALR field descriptions

Field	Description
IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

50.4.16 Descriptor Group Upper Address Register (GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Figure 867. Descriptor Group Upper Address Register (GAUR)

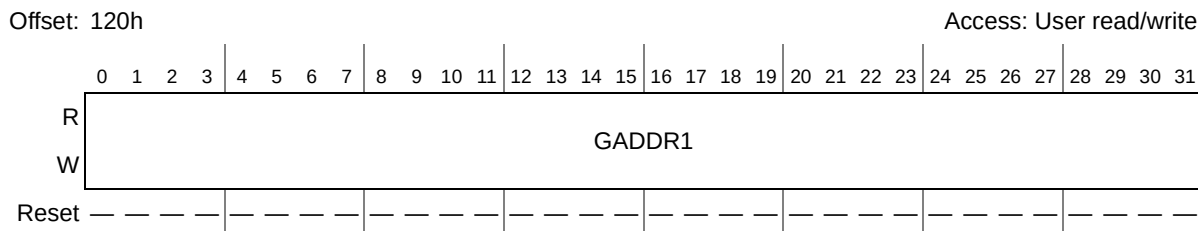


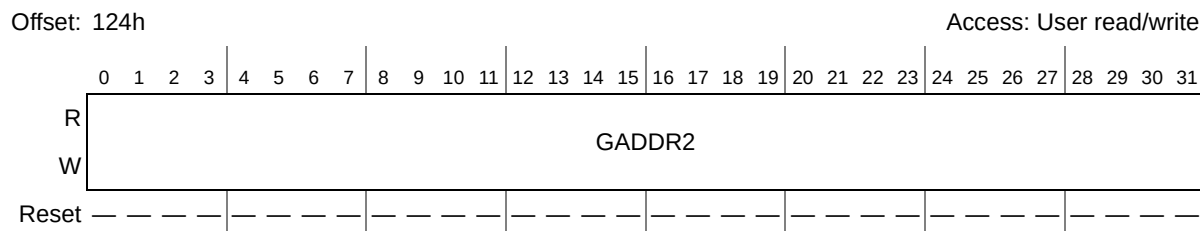
Table 895. GAUR field descriptions

Field	Description
GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

50.4.17 Descriptor Group Lower Address Register (GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

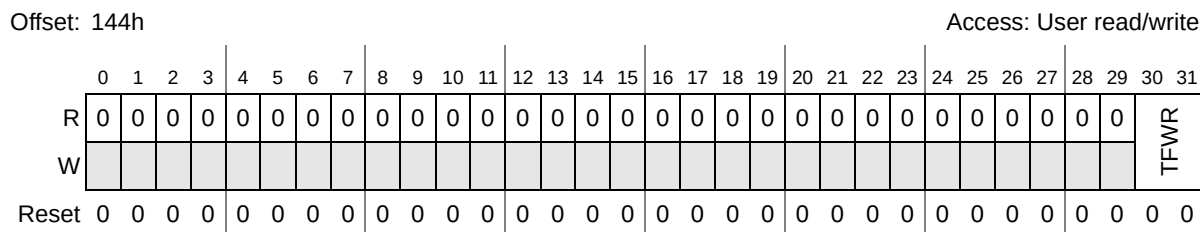
This register is not reset and you must initialize it.

Figure 868. Descriptor Group Lower Address Register (GALR)**Table 896. GALR field descriptions**

Field	Description
GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

50.4.18 Transmit FIFO Watermark Register (TFWR)

The TFWR controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

Figure 869. Transmit FIFO Watermark Register (TFWR)**Table 897. TFWR field descriptions**

Field	Description
TFWR	Number of bytes written to transmit FIFO before transmission of a frame begins 00 64 bytes written 01 64 bytes written 10 128 bytes written 11 192 bytes written

50.4.19 FIFO Receive Bound Register (FRBR)

FRBR indicates the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR, to appropriately divide the available FIFO RAM between the transmit and receive data paths.

Figure 870. FIFO Receive Bound Register (FRBR)

Offset: 14Ch

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	R_BOUND								0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Table 898. FRBR field descriptions

Field	Description
R_BOUND	Highest valid FIFO RAM address.

50.4.20 FIFO Receive Start Register (FRSR)

FRSR indicates the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

Hardware initializes the FRSR at reset. FRSR only needs to be written to change the default value.

Figure 871. FIFO Receive Start Register (FRSR)

Offset: 150h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	R_FSTART								0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

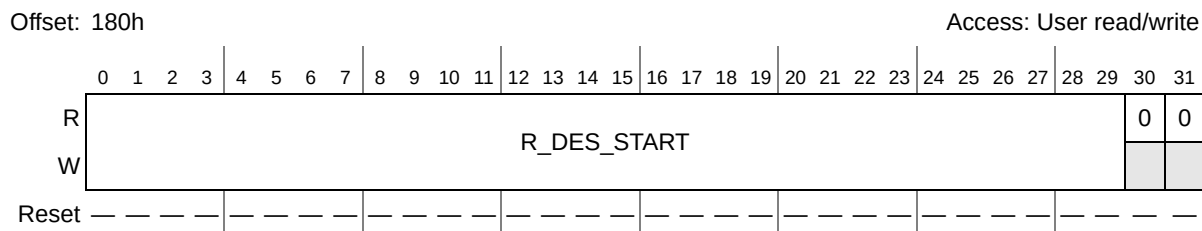
Table 899. FRSR field descriptions

Field	Description
R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs. For proper operation, ensure that R_FSTART is set to 48h or greater.

50.4.21 Receive Descriptor Ring Start Register (ERDSR)

ERDSR points to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

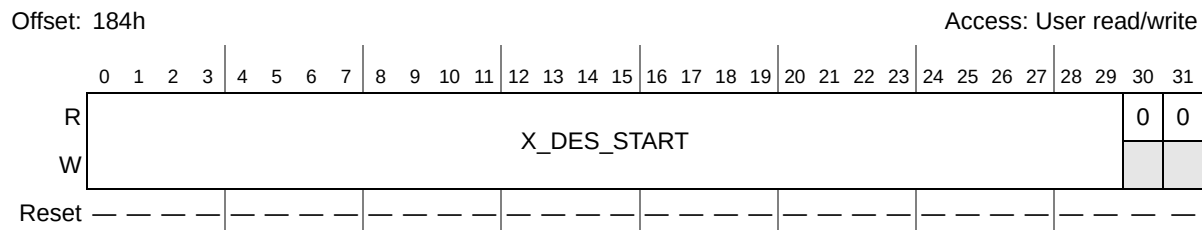
Figure 872. Receive Descriptor Ring Start Register (ERDSR)**Table 900. ERDSR field descriptions**

Field	Description
R_DES_START	Pointer to start of receive buffer descriptor queue.

50.4.22 Transmit Buffer Descriptor Ring Start Register (ETSDR)

ETSDR provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Figure 873. Transmit Buffer Descriptor Ring Start Register (ETSDR)**Table 901. ETSDR field descriptions**

Field	Description
X_DES_START	Pointer to start of transmit buffer descriptor queue.

50.4.23 Receive Buffer Size Register (EMRBR)

The EMRBR dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. To properly align the buffer, EMRBR must be evenly divisible by 16. To ensure this, bits 3–0 are forced low.

To minimize bus utilization (descriptor fetches), it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR register is undefined at reset and must be initialized by the user.

Figure 874. Receive Buffer Size Register (EMRBR)

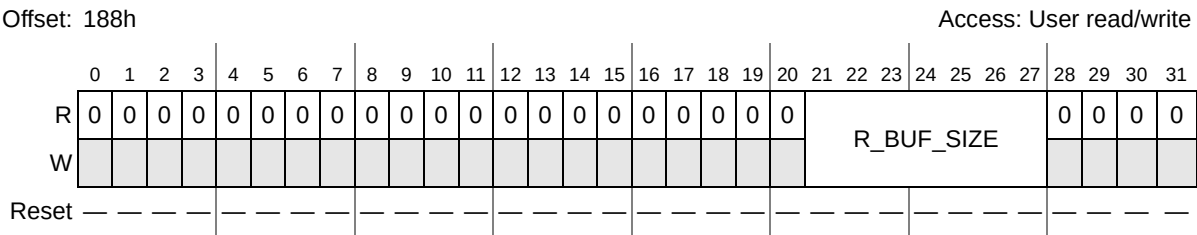


Table 902. EMRBR field descriptions

Field	Description
R_BUF_SIZE	Maximum size of receive buffer size in bytes. To minimize bus utilization (descriptor fetches), set this field to 256 bytes (10h) or larger. 10h 256 + 15 bytes (minimum size recommended) 11h 272 + 15 bytes ... 7Fh 2032 + 15 bytes. The FEC writes up to 2047 bytes in the receive buffer. If data larger than 2047 is received, the FEC truncates it and shows 7FFh in the receive descriptor

50.4.24 Count of frames not counted correctly (RMON_T_DROP)

Figure 875. Count of frames not counted correctly (RMON_T_DROP)

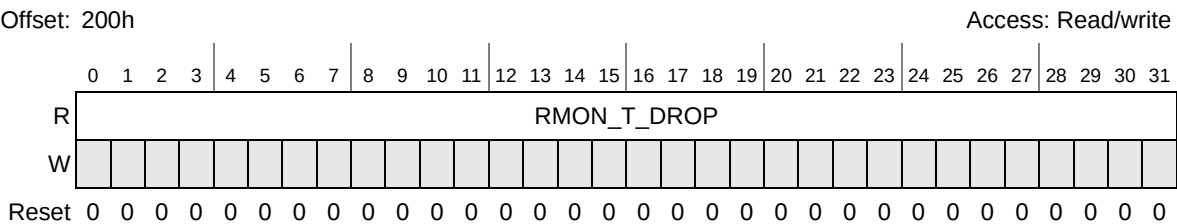


Table 903. RMON_T_DROP field descriptions

Field	Description
RMON_T_DROP	Count of frames not counted correctly

50.4.25 RMON Tx packet count (RMON_T_PACKETS)

Figure 876. RMON Tx packet count (RMON_T_PACKETS)

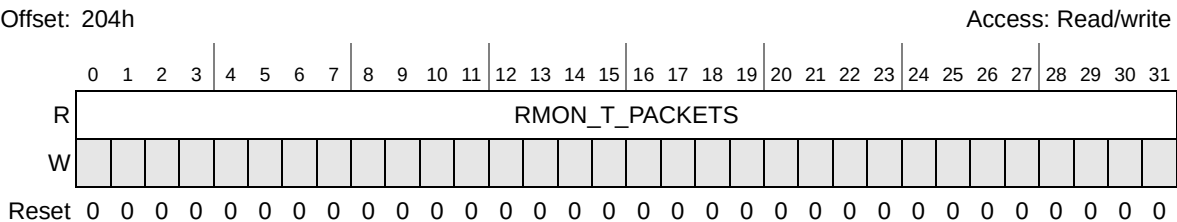


Table 904. RMON_T_PACKETS field descriptions

Field	Description
RMON_T_PACKETS	RMON Tx packet count

50.4.26 RMON Tx broadcast packets (RMON_T_BC_PKT)

Figure 877. RMON Tx broadcast packets (RMON_T_BC_PKT)

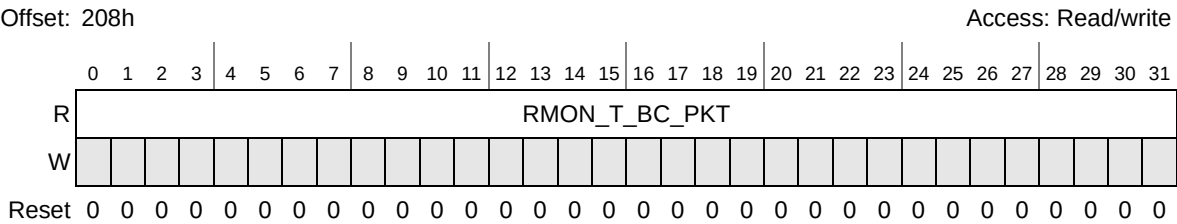


Table 905. RMON_T_PACKETS field descriptions

Field	Description
RMON_T_BC_PKT	RMON Tx broadcast packets

50.4.27 RMON Tx multicast packets (RMON_T_MC_PKT)

Figure 878. RMON Tx multicast packets (RMON_T_MC_PKT)

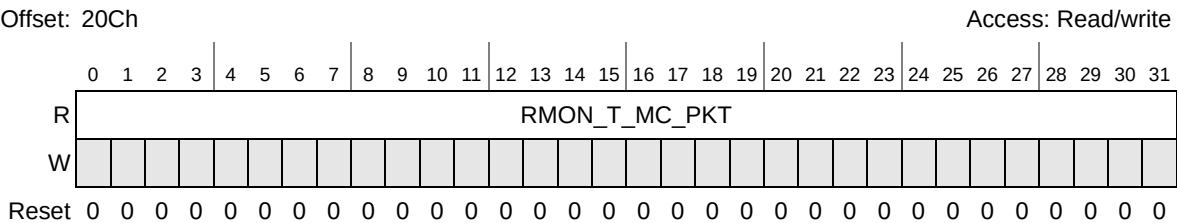


Table 906. RMON_T_MC_PKT field descriptions

Field	Description
RMON_T_MC_PKT	RMON Tx multicast packets

50.4.28 RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN)

Figure 879. RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN)

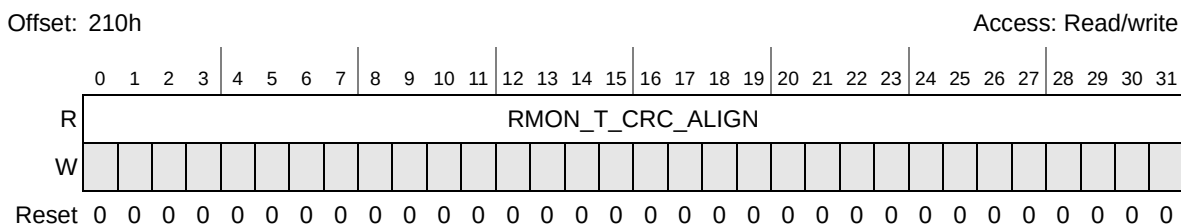


Table 907. RMON_T_CRC_ALIGN field descriptions

Field	Description
RMON_T_CRC_ALIGN	RMON Tx packets with CRC/align error

50.4.29 RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)

Figure 880. RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)

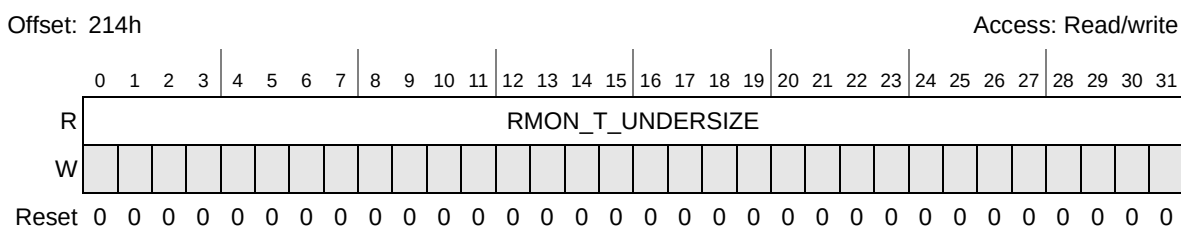


Table 908. RMON_T_CRC_ALIGN field descriptions

Field	Description
RMON_T_UNDERSIZE	RMON Tx packets < 64 bytes, good CRC

50.4.30 RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)

Figure 881. RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)

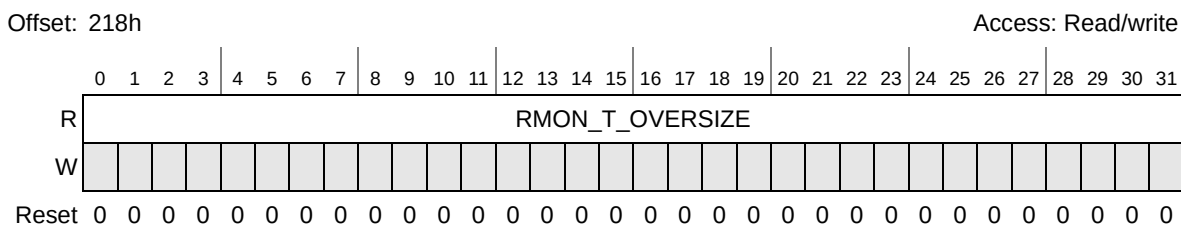


Table 909. RMON_T_OVERSIZE field descriptions

Field	Description
RMON_T_OVERSIZE	RMON Tx packets > MAX_FL bytes, good CRC

50.4.31 RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)

Figure 882. RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)

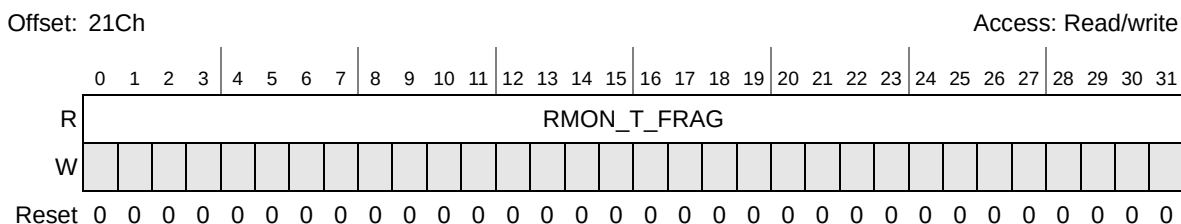


Table 910. RMON_T_FRAG field descriptions

Field	Description
RMON_T_FRAG	RMON Tx packets < 64 bytes, bad CRC

50.4.32 RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB)

Figure 883. RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB)

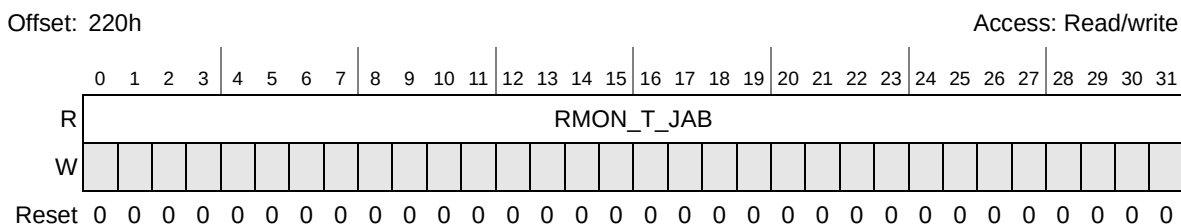


Table 911. RMON_T_JAB field descriptions

Field	Description
RMON_T_JAB	RMON Tx packets > MAX_FL bytes, bad CRC

50.4.33 RMON Tx collision count (RMON_T_COL)

Figure 884. RMON Tx collision count (RMON_T_COL)

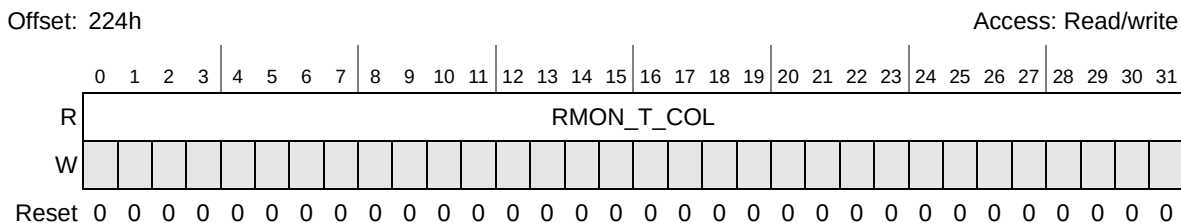


Table 912. RMON_T_COL field descriptions

Field	Description
RMON_T_COL	RMON Tx collision count

50.4.34 RMON Tx 64 byte packets (RMON_T_P64)

Figure 885. RMON Tx 64 byte packets (RMON_T_P64)

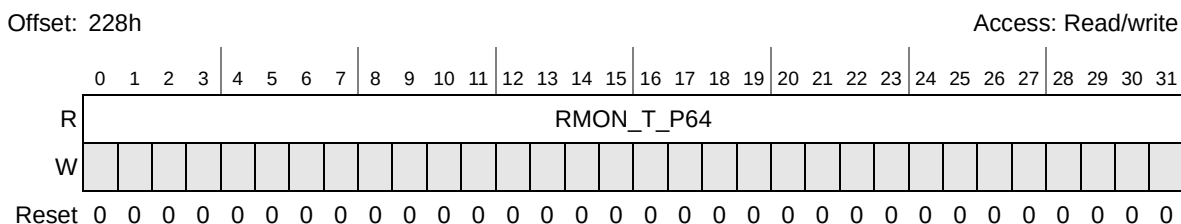


Table 913. RMON_T_P64 field descriptions

Field	Description
RMON_T_P64	RMON Tx 64 byte packets

50.4.35 RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)

Figure 886. RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)

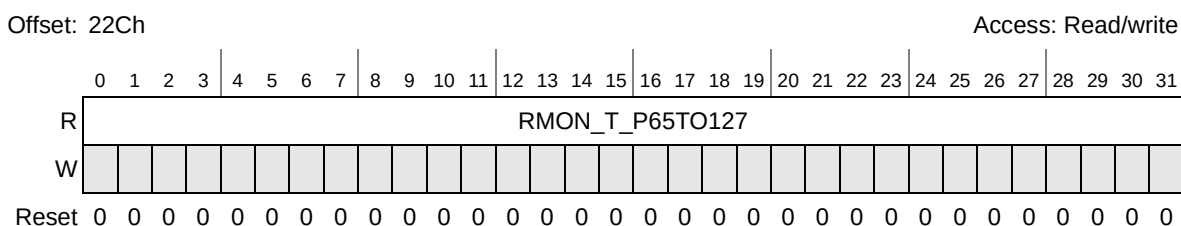


Table 914. RMON_T_P65TO127 field descriptions

Field	Description
RMON_T_P65TO127	RMON Tx 65 to 127 byte packets

50.4.36 RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)

Figure 887. RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)

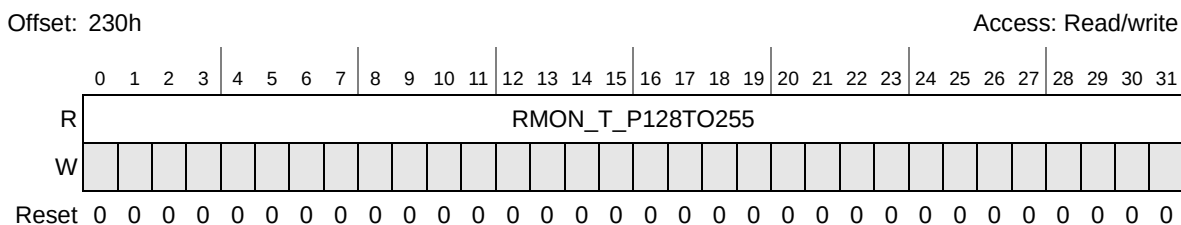


Table 915. RMON_T_P128TO255 field descriptions

Field	Description
RMON_T_P128TO255	RMON Tx 128 to 255 byte packets

50.4.37 RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)

Figure 888. RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)

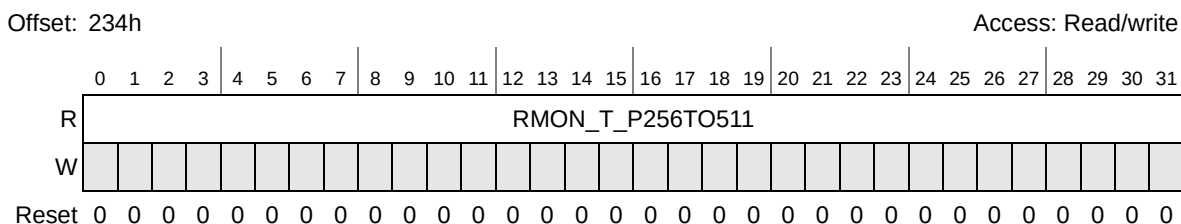


Table 916. RMON_T_P256TO511 field descriptions

Field	Description
RMON_T_P256TO511	RMON Tx 256 to 511 byte packets

50.4.38 RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)

Figure 889. RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)

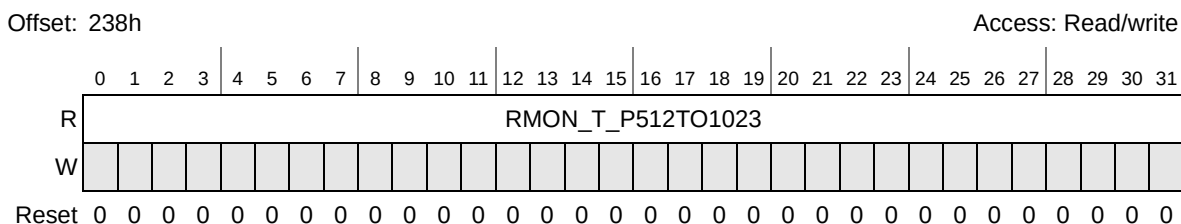


Table 917. RMON_T_P512TO1023 field descriptions

Field	Description
RMON_T_P512TO1023	RMON Tx 512 to 1023 byte packets

50.4.39 RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047)

Figure 890. RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047)

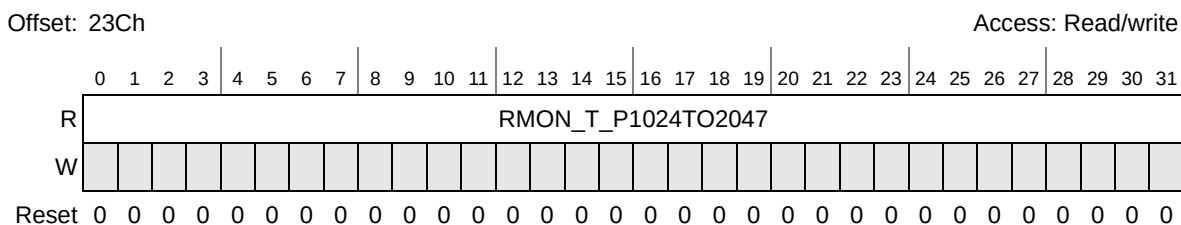


Table 918. RMON_T_P1024TO2047 field descriptions

Field	Description
RMON_T_P1024TO2047	RMON Tx 1024 to 2047 byte packets

50.4.40 RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)

Figure 891. RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)

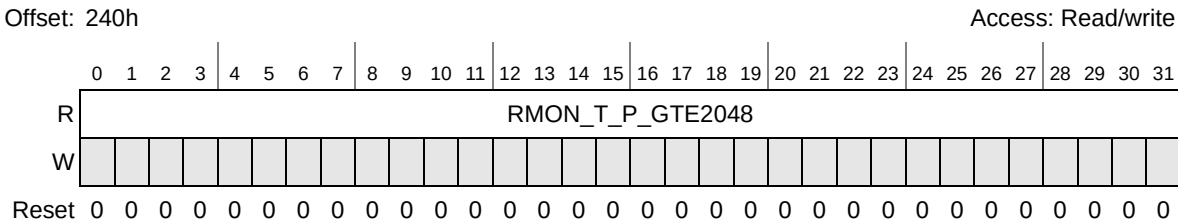


Table 919. RMON_T_P_GTE2048 field descriptions

Field	Description
RMON_T_P_GTE2048	RMON Tx packets with > 2048 bytes

50.4.41 RMON Tx Octets (RMON_T_OCTETS)

Figure 892. RMON Tx Octets (RMON_T_OCTETS)

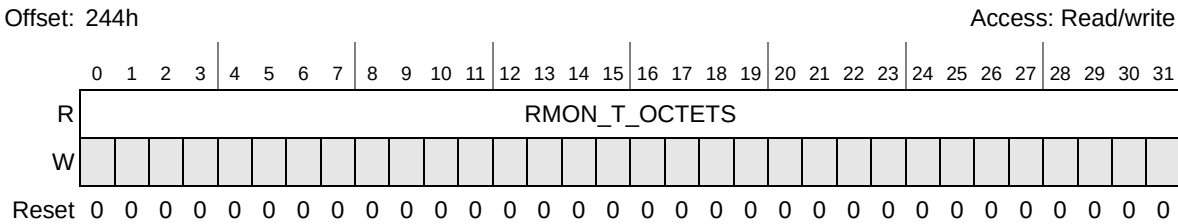


Table 920. RMON_T_OCTETS field descriptions

Field	Description
RMON_T_OCTETS	RMON Tx Octets

50.4.42 Count of transmitted frames not counted correctly (IEEE_T_DROP)

Figure 893. Count of transmitted frames not counted correctly (IEEE_T_DROP)

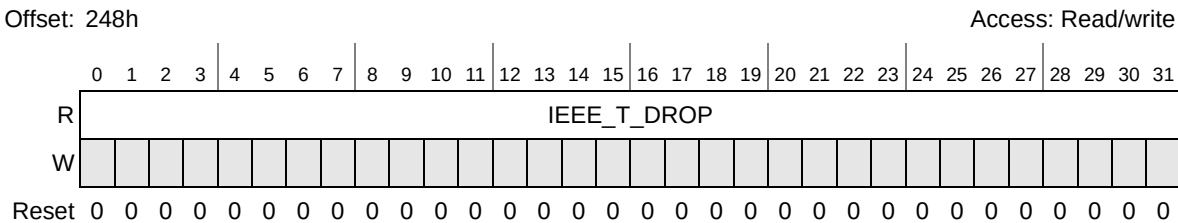


Table 921. IEEE_T_DROP field descriptions

Field	Description
IEEE_T_DROP	Count of transmitted frames not counted correctly

50.4.43 Frames transmitted OK (IEEE_T_FRAME_OK)

Figure 894. Frames transmitted OK (IEEE_T_FRAME_OK)

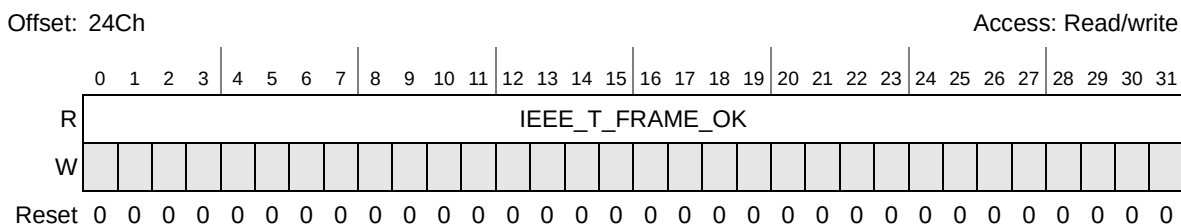


Table 922. IEEE_T_FRAME_OK field descriptions

Field	Description
IEEE_T_FRAME_OK	Frames transmitted OK

50.4.44 Frames transmitted with single collision (IEEE_T_1COL)

Figure 895. Frames transmitted with single collision (IEEE_T_1COL)

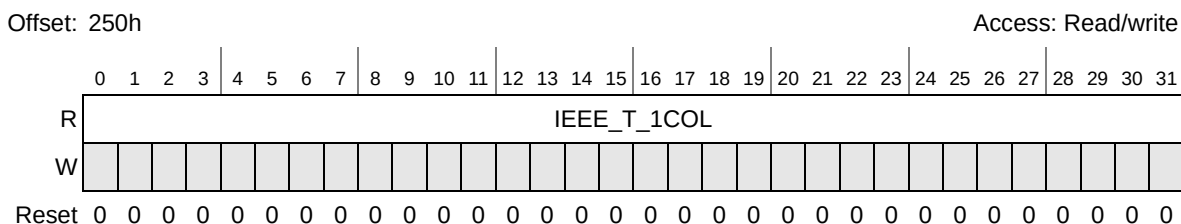


Table 923. IEEE_T_1COL field descriptions

Field	Description
IEEE_T_1COL	Frames transmitted with single collision

50.4.45 Frames transmitted with multiple collisions (IEEE_T_MCOL)

Figure 896. Frames transmitted with multiple collisions (IEEE_T_MCOL)

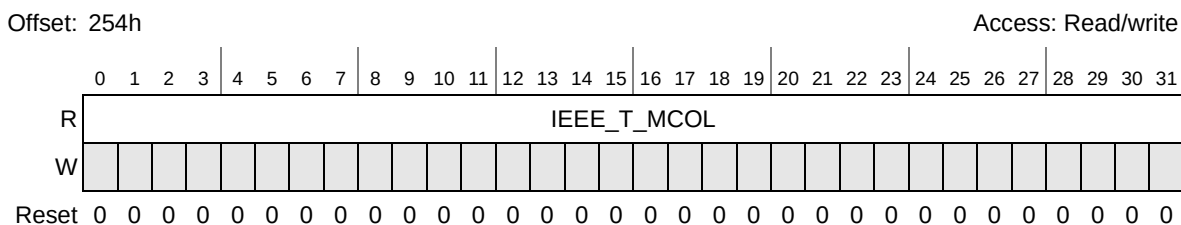


Table 924. IEEE_T_MCOL field descriptions

Field	Description
IEEE_T_MCOL	Frames transmitted with multiple collisions

50.4.46 Frames transmitted after deferral delay (IEEE_T_DEF)

Figure 897. Frames transmitted after deferral delay (IEEE_T_DEF)

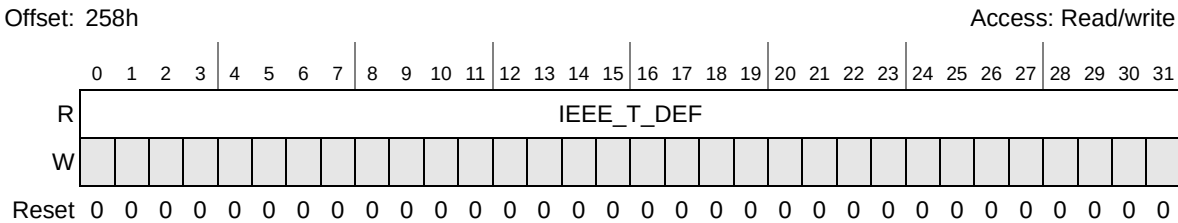


Table 925. IEEE_T_DEF field descriptions

Field	Description
IEEE_T_DEF	Frames transmitted after deferral delay

50.4.47 Frames transmitted with late collision (IEEE_T_LCOL)

Figure 898. Frames transmitted with late collision (IEEE_T_LCOL)

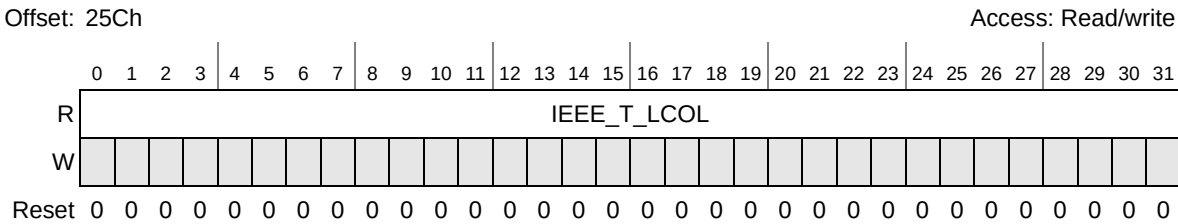


Table 926. IEEE_T_LCOL field descriptions

Field	Description
IEEE_T_LCOL	Frames transmitted with late collision

50.4.48 Frames transmitted with excessive collisions (IEEE_T_EXCOL)

Figure 899. Frames transmitted with excessive collisions (IEEE_T_EXCOL)

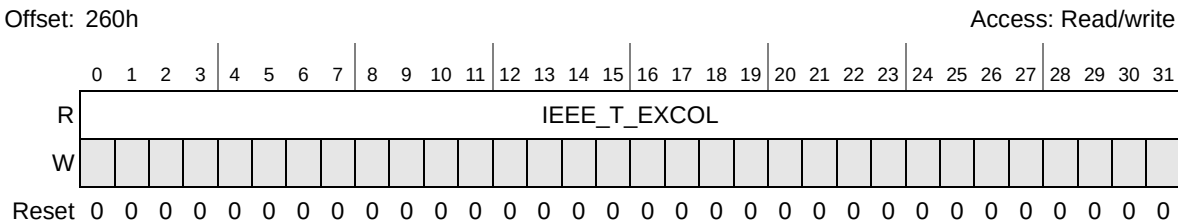


Table 927. IEEE_T_EXCOL field descriptions

Field	Description
IEEE_T_EXCOL	Frames transmitted with excessive collisions

50.4.49 Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR)

Figure 900. Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR)

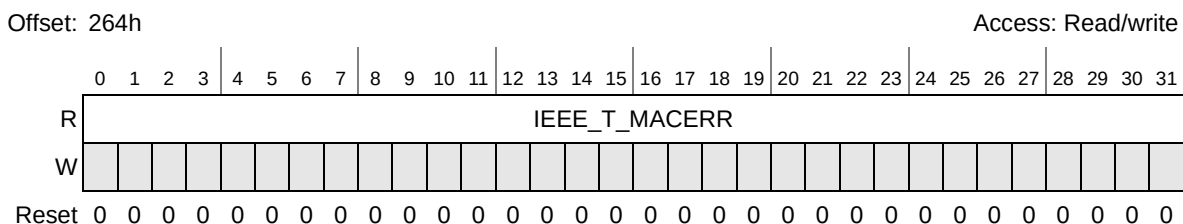


Table 928. IEEE_T_MACERR field descriptions

Field	Description
IEEE_T_MACERR	Frames transmitted with Tx FIFO underrun

50.4.50 Frames transmitted with carrier sense error (IEEE_T_CSERR)

Figure 901. Frames transmitted with carrier sense error (IEEE_T_CSERR)

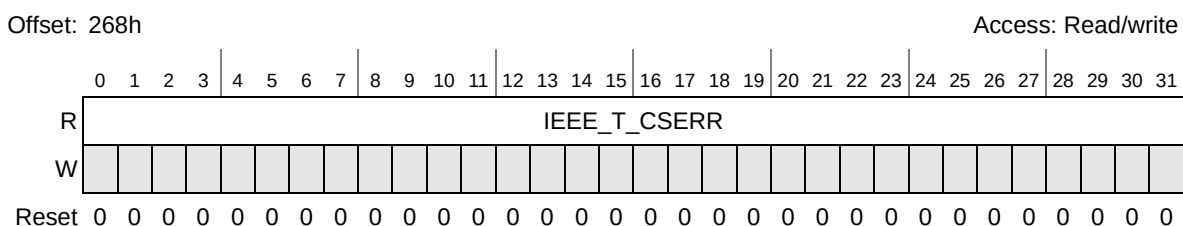


Table 929. IEEE_T_CSERR field descriptions

Field	Description
IEEE_T_CSERR	Frames transmitted with carrier sense error

50.4.51 Frames transmitted with SQE error (IEEE_T_SQE)

Figure 902. Frames transmitted with SQE error (IEEE_T_SQE)

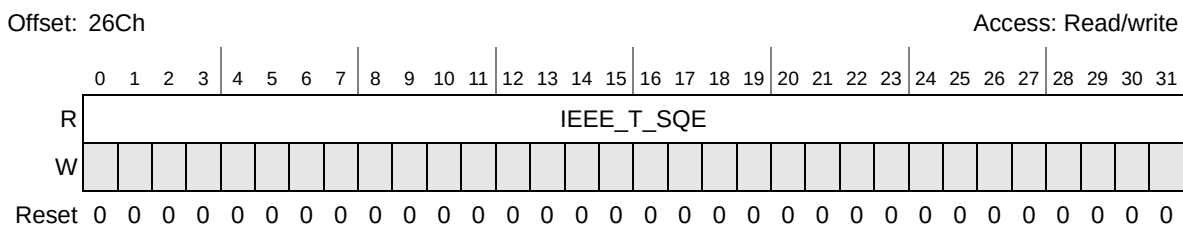


Table 930. IEEE_T_SQE field descriptions

Field	Description
IEEE_T_SQE	Frames transmitted with SQE error

50.4.52 Flow control pause frames transmitted (IEEE_T_FDXFC)

Figure 903. Flow control pause frames transmitted (IEEE_T_FDXFC)

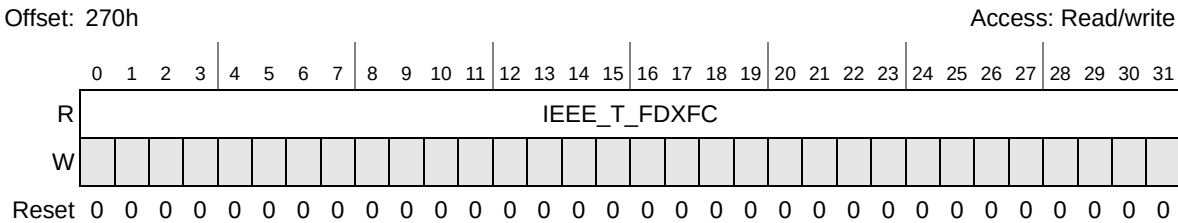


Table 931. IEEE_T_FDXFC field descriptions

Field	Description
IEEE_T_FDXFC	Flow control pause frames transmitted

50.4.53 Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)

Figure 904. Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)

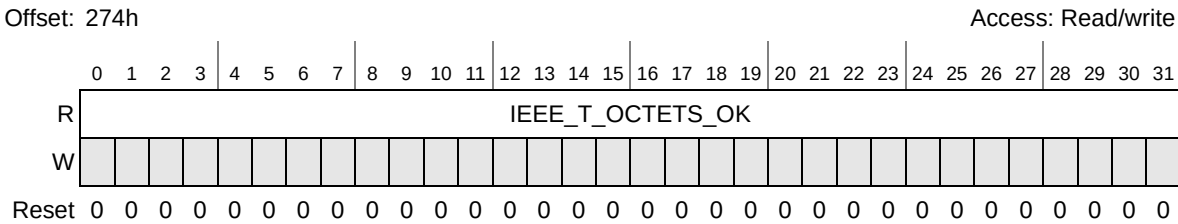


Table 932. IEEE_T_OCTETS_OK field descriptions

Field	Description
IEEE_T_OCTETS_OK	Octet count for frames transmitted without error

50.4.54 Count of received frames not counted correctly (RMON_R_DROP)

Figure 905. Count of received frames not counted correctly (RMON_R_DROP)

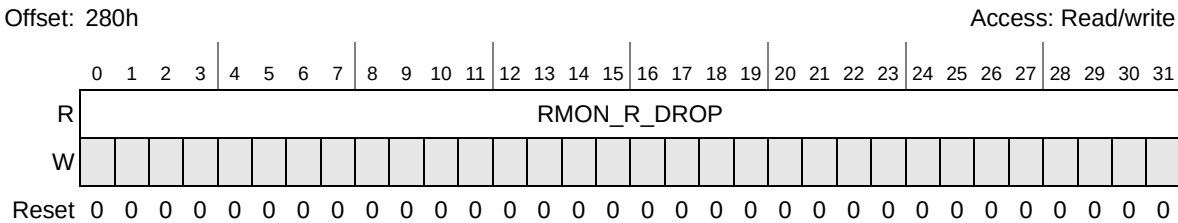


Table 933. RMON_R_DROP field descriptions

Field	Description
RMON_R_DROP	Count of received frames not counted correctly

50.4.55 RMON Rx packet count (RMON_R_PACKETS)

Figure 906. RMON Rx packet count (RMON_R_PACKETS)

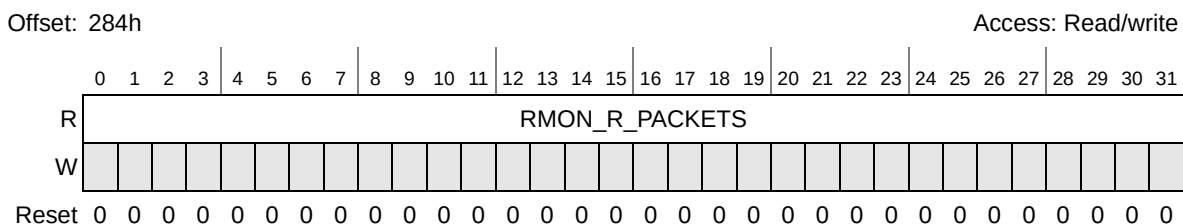


Table 934. RMON_R_PACKETS field descriptions

Field	Description
RMON_R_PACKETS	RMON Rx packet count

50.4.56 RMON Rx broadcast packets (RMON_R_BC_PKT)

Figure 907. RMON Rx broadcast packets (RMON_R_BC_PKT)

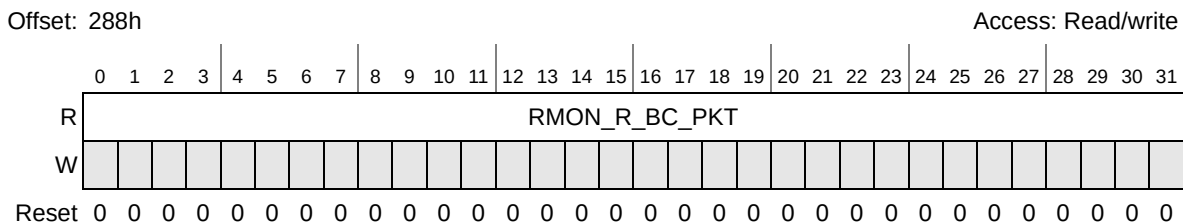


Table 935. RMON_R_BC_PKT field descriptions

Field	Description
RMON_R_BC_PKT	RMON Rx broadcast packets

50.4.57 RMON Rx multicast packets (RMON_R_MC_PKT)

Figure 908. RMON Rx multicast packets (RMON_R_MC_PKT)

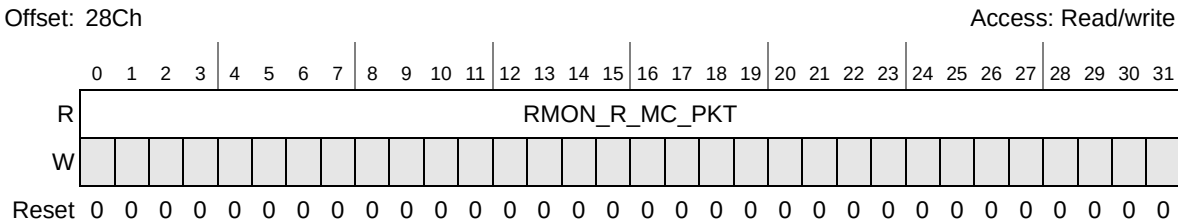


Table 936. RMON_R_MC_PKT field descriptions

Field	Description
RMON_R_MC_PKT	RMON Rx multicast packets

50.4.58 RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN)

Figure 909. RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN)

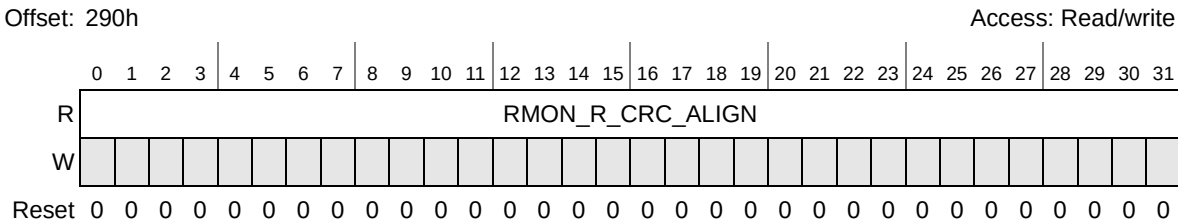


Table 937. RMON_R_CRC_ALIGN field descriptions

Field	Description
RMON_R_CRC_ALIGN	RMON Rx packets with CRC/Align error

50.4.59 RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)

Figure 910. RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)

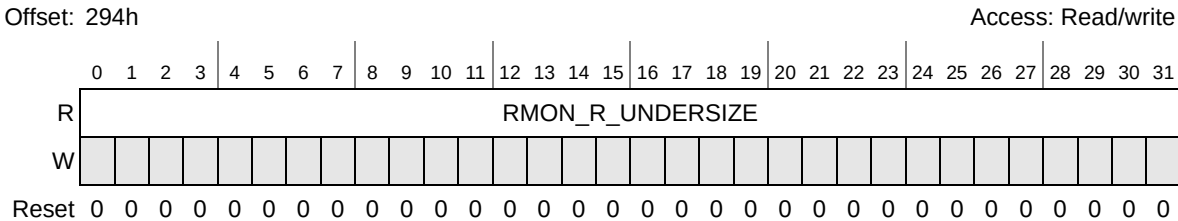


Table 938. RMON_R_UNDERSIZE field descriptions

Field	Description
RMON_R_UNDERSIZE	RMON Rx packets < 64 bytes, good CRC

50.4.60 RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)

Figure 911. RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)

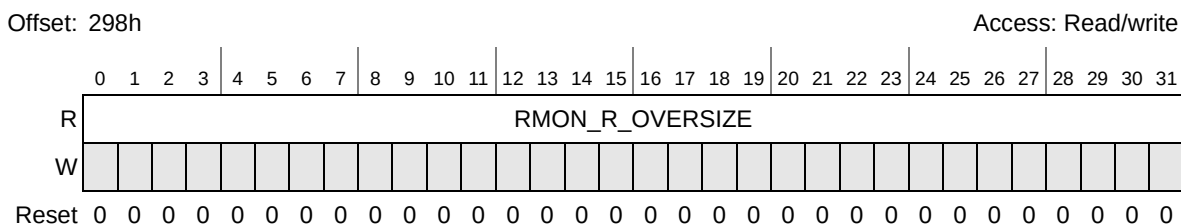


Table 939. RMON_R_OVERSIZE field descriptions

Field	Description
RMON_R_OVERSIZE	RMON Rx packets > MAX_FL bytes, good CRC

50.4.61 RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)

Figure 912. RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)

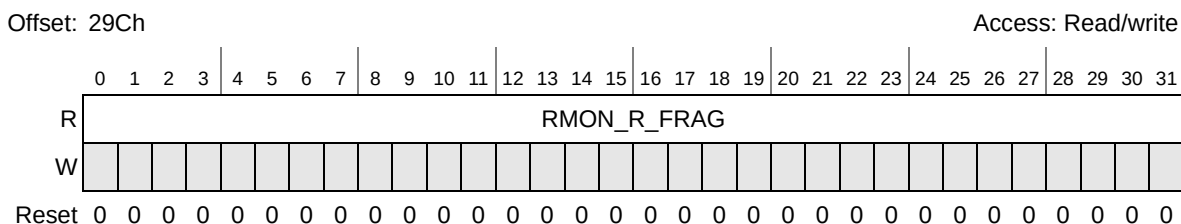


Table 940. RMON_R_FRAG field descriptions

Field	Description
RMON_R_FRAG	RMON Rx packets < 64 bytes, bad CRC

50.4.62 RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB)

Figure 913. RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB)

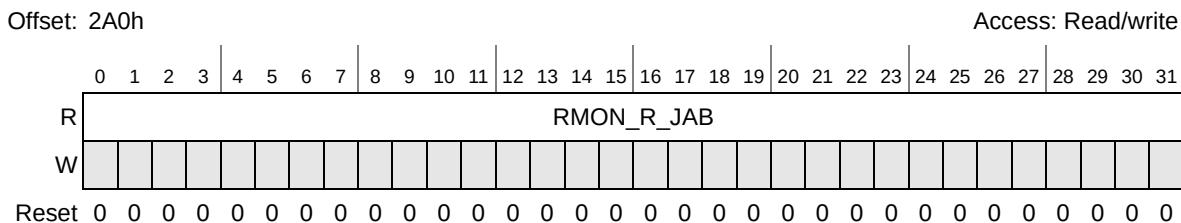


Table 941. RMON_R_JAB field descriptions

Field	Description
RMON_R_JAB	RMON Rx packets > MAX_FL bytes, bad CRC

50.4.63 Reserved (RMON_R_RESVD_0)

Figure 914. Reserved (RMON_R_RESVD_0)

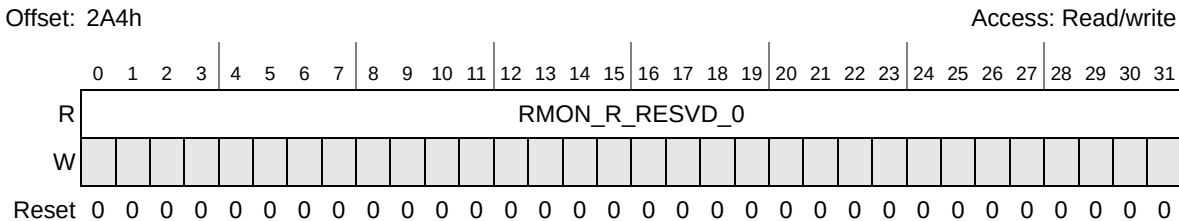


Table 942. RMON_R_RESVD_0 field descriptions

Field	Description
RMON_R_RESVD_0	Reserved

50.4.64 RMON Rx 64 byte packets (RMON_R_P64)

Figure 915. RMON Rx 64 byte packets (RMON_R_P64)

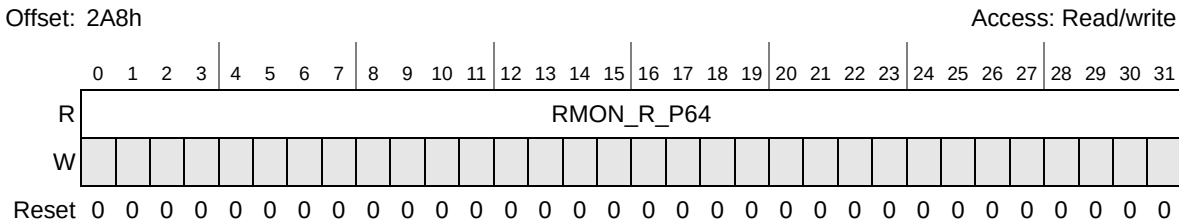


Table 943. RMON_R_P64 field descriptions

Field	Description
RMON_R_P64	RMON Rx 64 byte packets

50.4.65 RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)

Figure 916. RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)

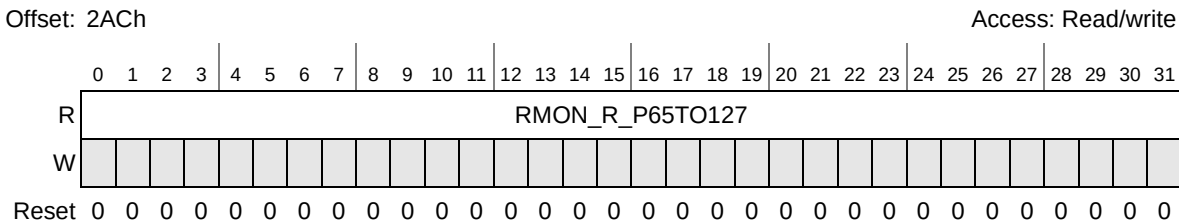


Table 944. RMON_R_P65TO127 field descriptions

Field	Description
RMON_R_P65TO127	RMON Rx 65 to 127 byte packets

50.4.66 RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)

Figure 917. RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)

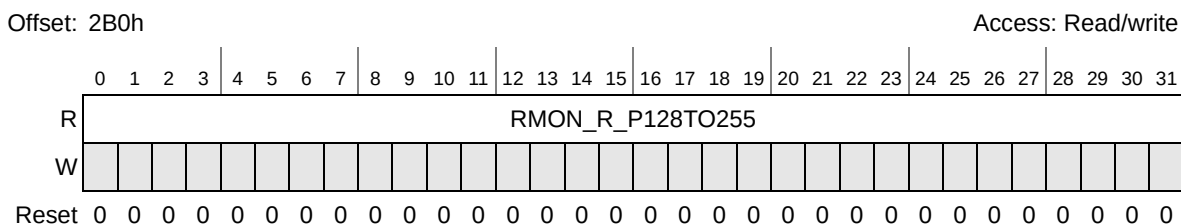


Table 945. RMON_R_P128TO255 field descriptions

Field	Description
RMON_R_P128TO255	RMON Rx 128 to 255 byte packets

50.4.67 RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)

Figure 918. RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)

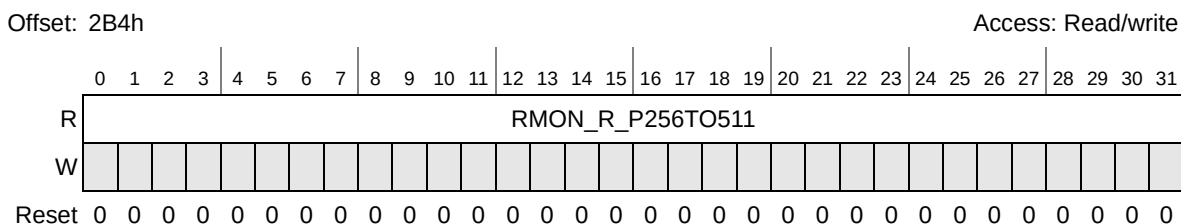


Table 946. RMON_R_P256TO511 field descriptions

Field	Description
RMON_R_P256TO511	RMON Rx 256 to 511 byte packets

50.4.68 RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023)

Figure 919. RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023)

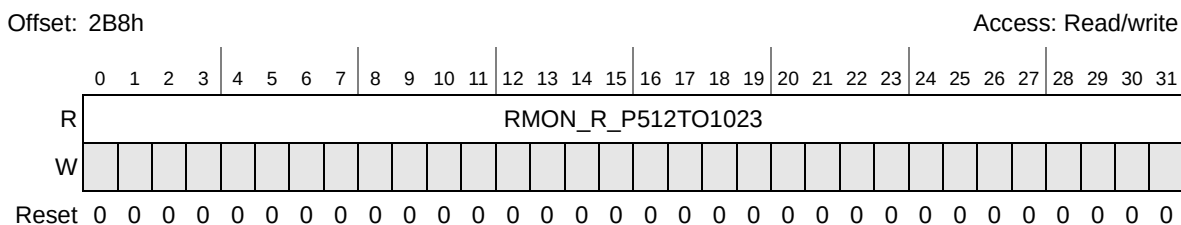


Table 947. RMON_R_P512TO1023 field descriptions

Field	Description
RMON_R_P512TO1023	RMON Rx 512 to 1023 byte packets

50.4.69 RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047)

Figure 920. RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047)

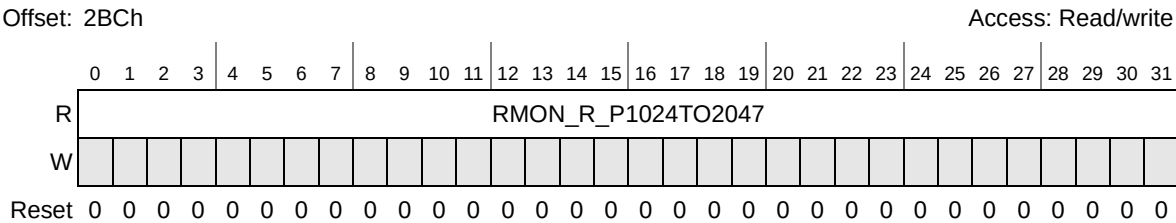


Table 948. RMON_R_P1024TO2047 field descriptions

Field	Description
RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets

50.4.70 RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048)

Figure 921. RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048)

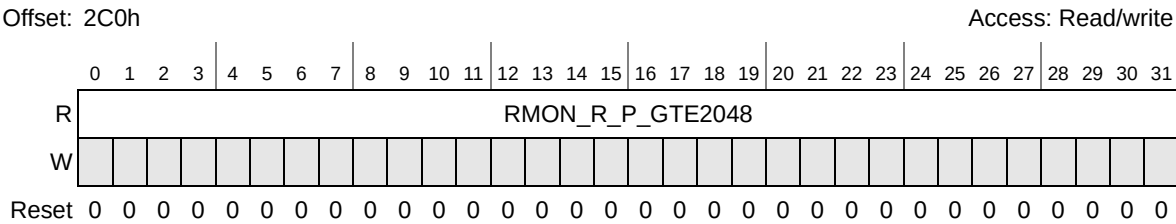


Table 949. RMON_R_P_GTE2048 field descriptions

Field	Description
RMON_R_P_GTE2048	RMON Rx packets with > 2048 bytes

50.4.71 RMON Rx octets (RMON_R_OCTETS)

Figure 922. RMON Rx octets (RMON_R_OCTETS)

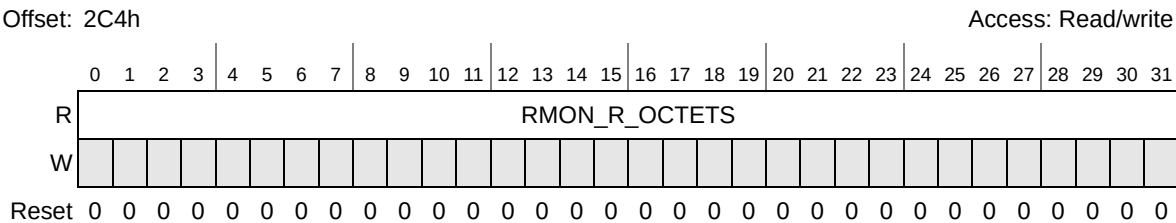


Table 950. RMON_R_OCTETS field descriptions

Field	Description
RMON_R_OCTETS	RMON Rx octets

50.4.72 Count of received frames not counted correctly (IEEE_R_DROP)

Figure 923. Count of received frames not counted correctly (IEEE_R_DROP)

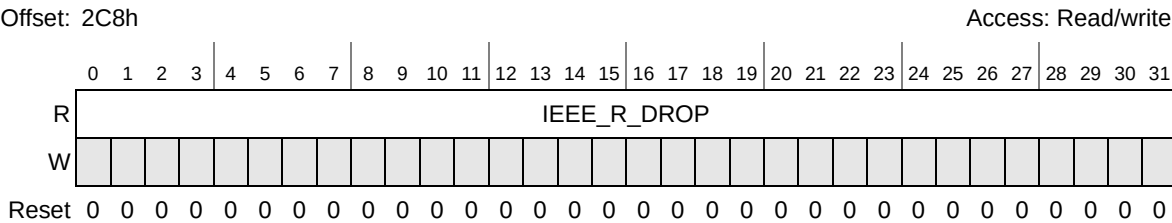


Table 951. IEEE_R_DROP field descriptions

Field	Description
IEEE_R_DROP	Count of received frames not counted correctly

50.4.73 Frames received OK (IEEE_R_FRAME_OK)

Figure 924. Frames received OK (IEEE_R_FRAME_OK)

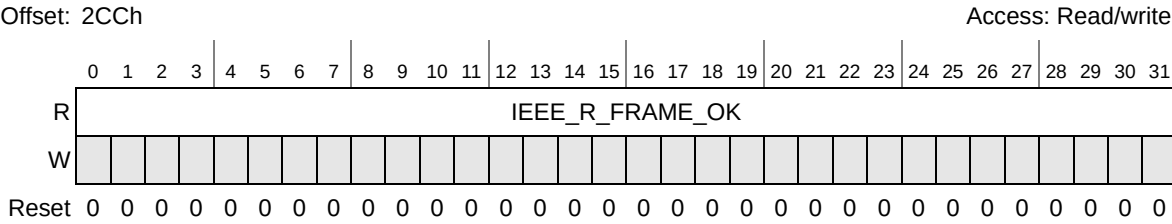


Table 952. IEEE_R_FRAME_OK field descriptions

Field	Description
IEEE_R_FRAME_OK	Frames received OK

50.4.74 Frames received with CRC error (IEEE_R_CRC)

Figure 925. Frames received with CRC error (IEEE_R_CRC)

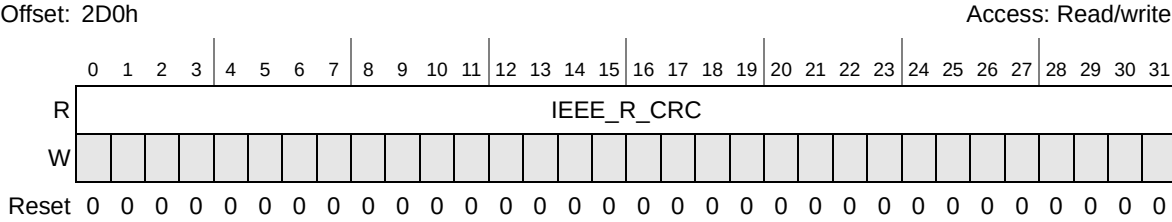


Table 953. IEEE_R_CRC field descriptions

Field	Description
IEEE_R_CRC	Frames received with CRC error

50.4.75 Frames received with alignment error (IEEE_R_ALIGN)

Figure 926. Frames received with alignment error (IEEE_R_ALIGN)

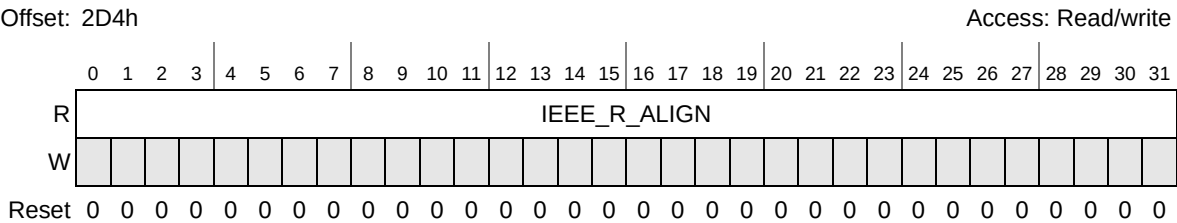


Table 954. IEEE_R_ALIGN field descriptions

Field	Description
IEEE_R_ALIGN	Frames received with alignment error

50.4.76 Receive FIFO overflow count (IEEE_R_MACERR)

Figure 927. Receive FIFO overflow count (IEEE_R_MACERR)

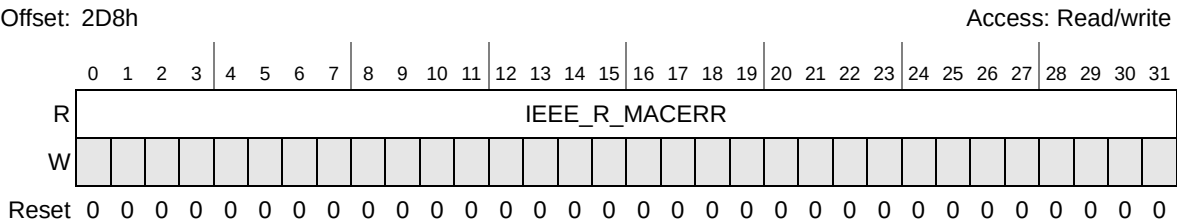


Table 955. IEEE_R_MACERR field descriptions

Field	Description
IEEE_R_MACERR	Receive FIFO overflow count

50.4.77 Flow control pause frames received (IEEE_R_FDXFC)

Figure 928. Flow control pause frames received (IEEE_R_FDXFC)

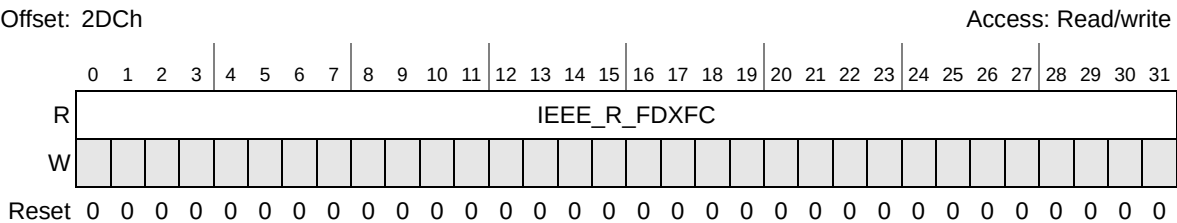


Table 956. IEEE_R_FDXFC field descriptions

Field	Description
IEEE_R_FDXFC	Flow control pause frames received

50.4.78 Octet count for frames received without error (IEEE_R_OCTETS_OK)

Figure 929. Octet count for frames received without error (IEEE_R_OCTETS_OK)

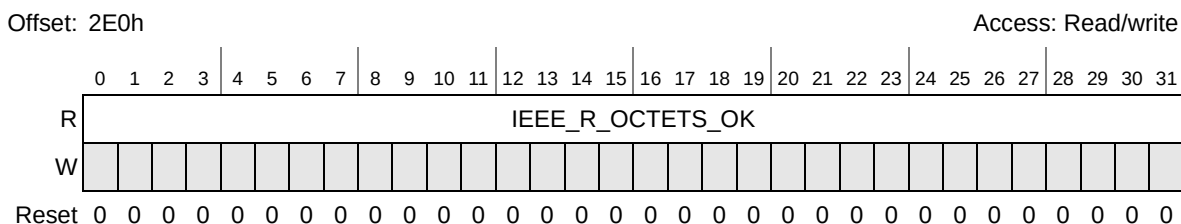


Table 957. IEEE_R_OCTETS_OK field descriptions

Field	Description
IEEE_R_OCTETS_OK	Octet count for frames received without error

50.5 Functional description

This section describes the operation of the FEC, beginning with the description of two different types of MII frames, the buffer descriptors, the hardware and software initialization sequence, then the software (Ethernet driver) interface for transmitting and receiving frames.

Following the software initialization and operation sections are sections providing a detailed description of the functions of the FEC.

50.5.1 MII data frame

Ethernet/802.3 data frames transmitted across the MII have the format specified in [Figure 930](#) and [Table 958](#).

Figure 930. MII data frame format

<inter-frame> <preamble> <sfd> <data> <efd>

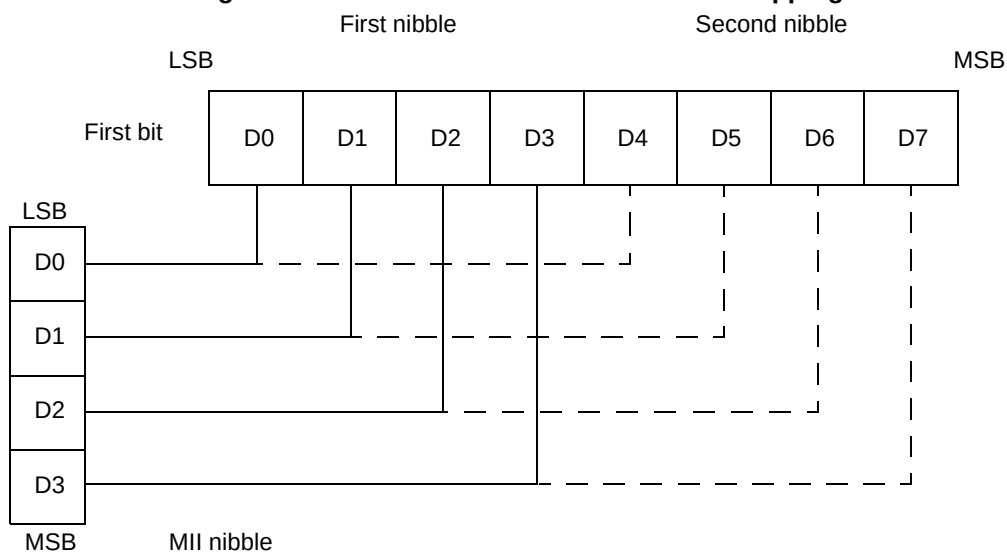
Table 958. MII data frame components

Component	Description
<inter-frame>	Inter-frame period. This is an unspecified amount of time during which no data activity occurs on the MII. The de-assertion of RX_DV and TX_EN indicate the absence of data activity.
<preamble>	Preamble that begins a frame. It has the following value: 10101010 10101010 10101010 10101010 10101010 10101010 10101010 The leftmost 1 represents the LSB of the byte.
<sfd>	Start of a frame. It has the value 10101011.

Table 958. MII data frame components (continued)

Component	Description
<data>	Data portion of the frame. It consists of N octets which corresponds to 2N nibbles being transmitted. The order of each nibble is defined in Figure 931 .
<efd>	End-of-frame delimiter. For TXD data, it is indicated by the de-assertion of the TX_EN signal. For RXD data, it is indicated by the de-assertion of the RX_DV signal.

Figure 931. MII nibble/octet to octet/nibble mapping



50.5.2 MII management frame structure

A transceiver management frame being transmitted on the MII management interface uses the MDIO and MDC pins. A transaction or frame on this serial interface has the format specified in [Figure 932](#) and [Table 959](#).

Figure 932. MII management frame format

<preamble> <st> <op> <phyad> <regad> <ta> <data> <idle>

Table 959. MII management frame components

Component	Description
<preamble>	Optional preamble, consisting of a sequence of 32 continuous logic 1's.
<st>	Start of frame, consisting of a 01 pattern.
<op>	Operation code. 01 Write operation 10 Read instruction
<phyad>	Five-bit field that allows up to 32 PHYs to be addressed. The first address bit transmitted is the MSB of the address.

Table 959. MII management frame components (continued)

Component	Description
<regad>	Five-bit field that allows for 32 registers to be addressed within each PHY. The first register bit transmitted is the MSB of the address.
<ta>	Two-bit field that provides spacing between the register address field and the data field to avoid contention on the MDIO signal during a read operation.
<data>	16-bit data field. Bit 15 is the first data bit transmitted and received.
<idle>	Idle condition (MDIO is in the high-impedance state).

The MII management register set located in the PHY may consist of a basic register set and an extended register set as defined in [Table 960](#).

Table 960. MII management register set

Register address	Register name	Register set
0	Control	Basic
1	Status	
2–3	PHY identifier	Extended
4	Auto-negotiation advertisement	
5	AN link partner ability	
6	AN expansion	
7	AN next page transmit	
8–15	Reserved	
16–31	Vendor-specific	

50.5.3 Buffer descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors. It is followed by a detailed description of the receive and transmit descriptor fields.

50.5.3.1 Driver/DMA operation with buffer descriptors

The data for the FEC frames resides in one or more memory buffers external to the FEC. Associated with each buffer is a buffer descriptor (BD), which contains a starting address (32-bit aligned pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read by the FEC DMA engine.

Software produces buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit produces the buffer. Software writing to TDAR or RDAR tells the FEC that a buffer is placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and consumes the buffers after they have been produced. After the data DMA is complete and the DMA engine writes the buffer descriptor status bits, hardware clears RxBD[E] or TxBD[R] to signal the buffer has been consumed. Software may poll the BDs to detect when the buffers are consumed.

or may rely on the buffer/frame interrupts. The driver may process these buffers, and they can return to the free list.

The ECR[ETHER_EN] bit operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is cleared, the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before ECR[ETHER_EN] is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The wrap (W) bit defines the last buffer descriptor in each ring. When W is set, the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively. Buffer descriptor rings must start on a 32-bit boundary; however, it is recommended they be 128-bit aligned.

50.5.3.1.1 Driver/DMA operation with transmit BDs

Typically, a transmit frame is divided between multiple buffers. An example is to have an application payload in one buffer, TCP header in a second buffer, IP header in a third buffer, and Ethernet/IEEE 802.3 header in a fourth buffer. The Ethernet MAC does not prepend the Ethernet header (destination address, source address, length/type field(s)), so the driver must provide this in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. TxBD[TC], which must be set by the driver, determines whether the MAC or driver appends the CRC.

The driver (TxBD software producer) should set up Tx BDs so a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length, and control (W, L, TC, ABC) and then the TxBD[R] bit should be set in reverse order (third, second, then first BD) to ensure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the second was made available, potentially causing a transmit FIFO underrun.

In the FEC, the driver notifies the DMA that new transmit frame(s) are available by writing to TDAR. When this register is written to (data value is not significant), the FEC RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers until a transmit BD is encountered with the R bit cleared. At this point, the FEC polls this BD one more time. If the R bit is cleared the second time, RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

50.5.3.1.2 Driver/DMA operation with receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore, the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) should set up some number of empty buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the L bit (1 indicates last buffer in frame), the frame status bits (if L is set), and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end-of-frame buffers, the receive BD is written with L set and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not only the length of the last buffer.

For simplicity, the driver may assign a large enough default receive buffer length to contain an entire frame, keeping in mind that a malfunction on the network or out-of-spec implementation could result in giant frames. Frames of 2K (2048) bytes or larger are truncated by the FEC at 2047 bytes so software never sees a receive frame larger than 2047 bytes.

Similar to transmit, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received, the FEC fills receive buffers and updates the associated BDs, then reads the next BD in the receive descriptor ring. If the FEC reads a receive BD and finds the E bit cleared, it polls this BD once more. If RxBD[E] is clear a second time, FEC stops reading receive BDs until the driver writes to RDAR.

50.5.3.2 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E and W bits in the first longword and the pointer in the second longword. When the buffer has been DMA'd, the Ethernet controller modifies the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first longword. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first longword of the buffer descriptor are only modified by the Ethernet controller when the L bit is set.

Figure 933. Receive Buffer Descriptor (RxBD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data Length															
Offset + 4	Rx Data Buffer Pointer - A[31:16]															
Offset + 6	Rx Data Buffer Pointer - A[15:0]															

Table 961. Receive Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	E	Empty. Written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.

Table 961. Receive Buffer Descriptor field descriptions (continued)

Word	Field	Description
Offset + 0	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ERDSR.
Offset + 0	RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	M	Miss. Written by the FEC. This bit is set by the FEC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
Offset + 0	BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).
Offset + 0	MC	Set if the DA is multicast and not BC.
Offset + 0	LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
Offset + 0	NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set, the CR bit is not set.
Offset + 0	CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
Offset + 0	OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and are zero. This bit is valid only if the L-bit is set.
Offset + 0	TR	Set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L equals 0 (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the FEC once as the BD is closed.
Offset + 4	A[31:16]	RX data buffer pointer, bits [31:16] ⁽¹⁾
Offset + 6	A[15:0]	RX data buffer pointer, bits [15:0]

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. The Ethernet controller never modifies this value.

Note: When the software driver sets an E bit in one or more receive descriptors, the driver should follow with a write to RDAR.

50.5.3.3 Ethernet Transmit Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. The Ethernet controller confirms transmission by clearing the ready bit (TxBD[R]) when DMA of the buffer is complete. In the TxBD, the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword and the buffer pointer in the second longword.

The FEC clears the R bit when the buffer is transferred. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block. See the memory map for more details.

Figure 934. Transmit Buffer Descriptor (TxBD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	R	TO1	W	TO2	L	TC	ABC	—	—	—	—	—	—	—	—	—
Offset + 2	Data Length															
Offset + 4	Tx Data Buffer Pointer - A[31:16]															
Offset + 6	Tx Data Buffer Pointer - A[15:0]															

Table 962. Transmit Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	R	Ready. Written by the FEC and you. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this bit is set.
Offset + 0	TO1	Transmit software ownership. This field is reserved for software use. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 0	TC	Transmit CRC. Written by user (only valid if L is set). 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte
Offset + 0	ABC	Append bad CRC. Written by user (only valid if L is set). 0 No effect 1 Transmit the CRC sequence inverted after the last data byte (regardless of TC value)

Table 962. Transmit Buffer Descriptor field descriptions (continued)

Word	Field	Description
Offset + 2	Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC.
Offset + 4	A[31:16]	Tx data buffer pointer, bits [31:16] ⁽¹⁾
Offset + 6	A[15:0]	Tx data buffer pointer, bits [15:0]

1. The transmit buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

Note: After the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame is setting the *R* bit in the first BD for the frame. The driver must follow that with a write to TDAR that triggers the FEC to poll the next BD in the ring.

50.5.4 Initialization sequence

This section describes which registers are reset due to hardware reset, which registers are reset by the FEC RISC, and what locations you must initialize prior to enabling the FEC.

50.5.4.1 Hardware Controlled Initialization

In the FEC, hardware resets registers and control logic that generate interrupts. A hardware reset negates output signals and resets general configuration bits.

Other registers reset when the ECR[ETHER_EN] bit is cleared (which is accomplished by a hard reset or software to halt operation). By clearing ECR[ETHER_EN], configuration control registers such as the TCR and RCR are not reset, but the entire data path is reset.

Table 963. ECR[ETHER_EN] De-Assertion Effect on FEC

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

50.5.5 User initialization (Prior to Setting ECR[ETHER_EN])

You need to initialize portions the FEC prior to setting the ECR[ETHER_EN] bit. The exact values depend on the particular application. The sequence is not important.

The following registers require initialization:

- EIMR
- EIR (clear by writing FFFF_FFFFh)
- TFWR (optional)
- IALR / IAUR
- GAUR / GALR
- PALR / PAUR (only needed for full duplex flow control)
- OPD (only needed for full duplex flow control)
- RCR
- TCR
- MSCR (optional)
- MIB_RAM (clear)
- FRSR (optional)
- EMRBR
- ERDSR
- ETDSR
- Transmit Descriptor ring (empty)
- Receive Descriptor ring (empty)

50.5.6 Microcontroller initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, hardware is ready for operation.

The sequence is as follows:

1. Initialize the backoff random number seed.
2. Activate receiver.
3. Activate transmitter.
4. Clear transmit FIFO.
5. Clear receive FIFO.
6. Initialize the transmit ring pointer.
7. Initialize the receive ring pointer.
8. Initialize FIFO count registers.

50.5.7 User initialization (after setting ECR[ETHER_EN])

After setting ECR[ETHER_EN], you can set up the buffer/frame descriptors and write to TDAR and RDAR. See [Section 50.5.3, Buffer descriptors](#), for more details.

50.5.8 Network interface options

The FEC supports an MII and reduced MII interface for 10/100 Mbps Ethernet, as well as a 7-wire serial interface for 10 Mbps Ethernet. The RCR[MII_MODE] and RCR[RMII_MODE] bits select the interface mode. In MII mode (RCR[MII_MODE] = 1 and RCR[RMII_MODE] = 0), there are 18 signals defined by the IEEE 802.3 standard and supported by the EMAC. [Table 964](#) shows these signals.

Table 964. MII mode

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[3:0]
Transmit Error	FEC_TXER
Collision	FEC_COL
Carrier Sense	FEC_CRS
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[3:0]
Receive Error	FEC_RXER
Management Data Clock	FEC_MDC
Management Data Input/Output	FEC_MDIO

In RMII mode ($\text{RCR}[\text{MII_MODE}] = 1$ and $\text{RCR}[\text{RMII_MODE}] = 1$), the EMAC supports 8 external signals. These signals are shown in [Table 965](#) below.

Table 965. RMII mode configuration

Signal description	EMAC pin
Reference Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[1:0]
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[1:0]
Receive Error	FEC_RXER

The 7-wire serial mode interface ($\text{RCR}[\text{MII_MODE}] = 1$ and $\text{RCR}[\text{RMII_MODE}] = 0$ or 1) is generally referred to as AMD mode. [Table 966](#) shows the 7-wire mode connections to the external transceiver.

Table 966. 7-Wire mode configuration

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[0]
Collision	FEC_COL

Table 966. 7-Wire mode configuration (continued)

Signal description	EMAC pin
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[0]

50.5.9 FEC frame transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ECR[ETHER_EN] is set and data appears in the transmit FIFO, the Ethernet MAC can transmit onto the network. The Ethernet controller transmits bytes least significant bit (lsb) first.

When the transmit FIFO fills to the watermark (defined by TFWR), MAC transmit logic asserts FEC_TXEN and starts transmitting the

(PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (FEC_CRSS is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive). See [Section 50.5.19.1, Transmission errors](#), for more details.

If a collision occurs during transmission of the frame (half duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data is transmitted, FCS (frame check sequence) or 32-bit cyclic redundancy check (CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Transmit logic automatically pads short frames (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set).

Settings in the EIMR determine interrupts generated to the buffer (TXB) and frame (TXF).

The transmit error interrupts are HBERR, BABT, LC, RL, and UN. If the transmit frame length exceeds MAX_FL bytes, BABT interrupt is asserted. However, the entire frame is transmitted (no truncation).

To pause transmission, set TCR[GTS] (graceful transmit stop). The FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame finishes or terminates with a collision. After the transmitter has stopped, the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

50.5.9.1 Duplicate frame transmission

The FEC fetches transmit buffer descriptors (TxBDs) and the corresponding transmit data continuously until the transmit FIFO is full. It does not determine whether the TxBD to be

fetches is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. To remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. It is possible that the FEC fetches from memory a BD that has already been processed but not yet written back (it is read a second time with the R bit remains set). In this case, the data is fetched and transmitted again.

Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for large or small frames, one of the following must be true:

- The FEC software driver ensures that there is always at least one TxBD with the ready bit cleared.
- Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched.
- The FEC software driver ensures a minimum frame size, n . The minimum number of TxBDs is then $(\text{Tx FIFO Size} \div (n + 4))$ rounded up to the nearest integer (though the result cannot be less than three). The default Tx FIFO size is 192 bytes; this size is programmable.

50.5.10 FEC frame reception

The FEC receiver works with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking. The Ethernet controller receives serial data lsb first.

When the driver enables the FEC receiver by setting ECR[ETHER_EN], it immediately starts processing receive frames. When FEC_RXDV is asserted, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the receiver processes the frame. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit times of RX_D0 following assertion of FEC_RXDV are ignored. Following the first 16 bit times, the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame are received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data is received and if address recognition has not rejected the frame, the receive FIFO signals the frame is accepted and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Therefore, no collision fragments are presented to you except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV, and TR status bits, and the frame length. See [Section 50.5.19.2, Reception errors](#), for more details.

Receive buffer (RXB) and frame interrupts (RFINT) may be generated if enabled by the EIMR register. A receive error interrupt is a babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR

interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See [Section 50.5.3.2, Ethernet Receive Buffer Descriptor \(RxBD\)](#), for more details.

When the receive frame is complete, the FEC sets the L-bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt (RFINT bit in EIR, maskable by RFIEN bit in EIMR), indicating that a frame is received and is in memory. The Ethernet controller then waits for a new frame.

50.5.11 Ethernet address recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames appears in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in [Figure 935](#) illustrates the address recognition decisions made by the receive block, while [Figure 936](#) illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in [Figure 935](#). Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in [Figure 936](#).

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame.

If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines the received frame is a valid PAUSE frame, the frame is rejected. The receiver detects a PAUSE frame with the DA field set to the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that you program in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR. In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on PAUSE frame detection, shown in [Figure 935](#).

If neither a hash match (group or individual) nor an exact match (group or individual) occur, and if promiscuous mode is enabled (RCR[PROM] set), the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.

Figure 935. Ethernet address recognition—receive block decisions

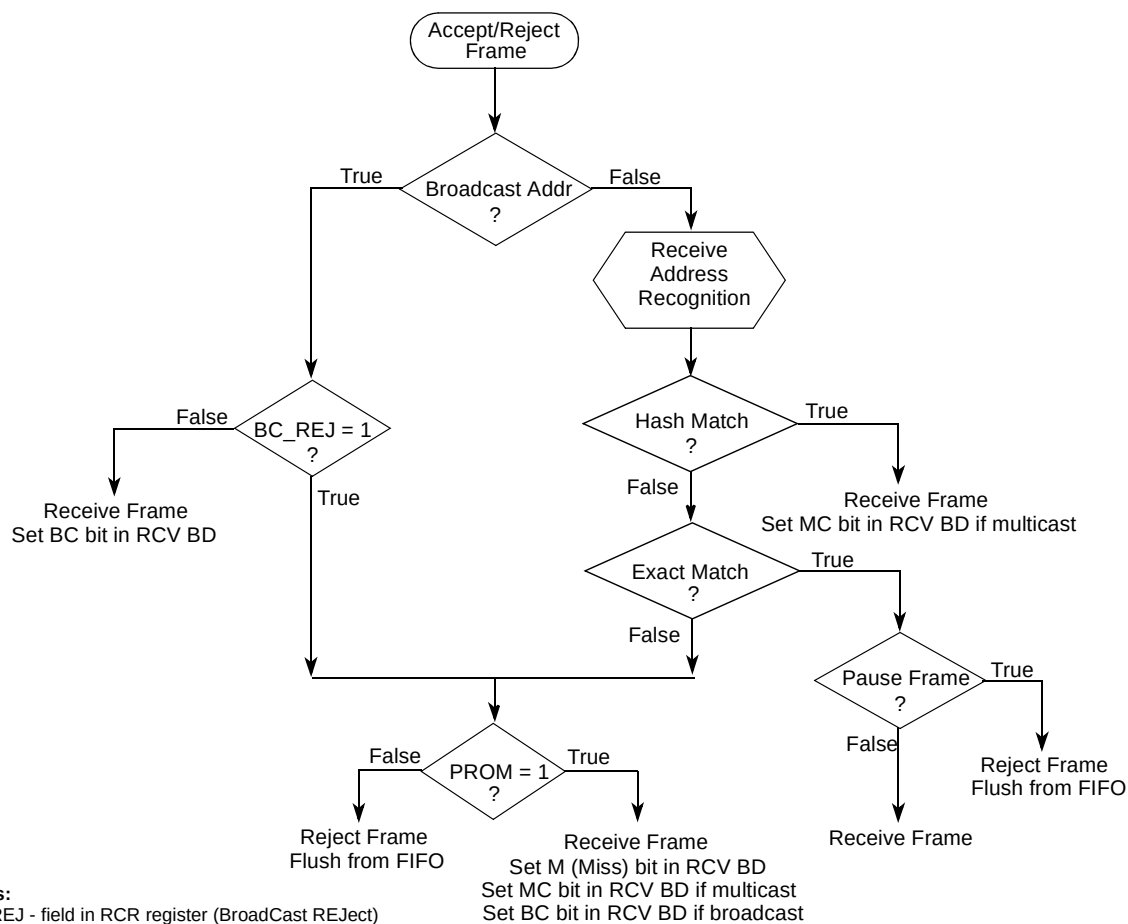
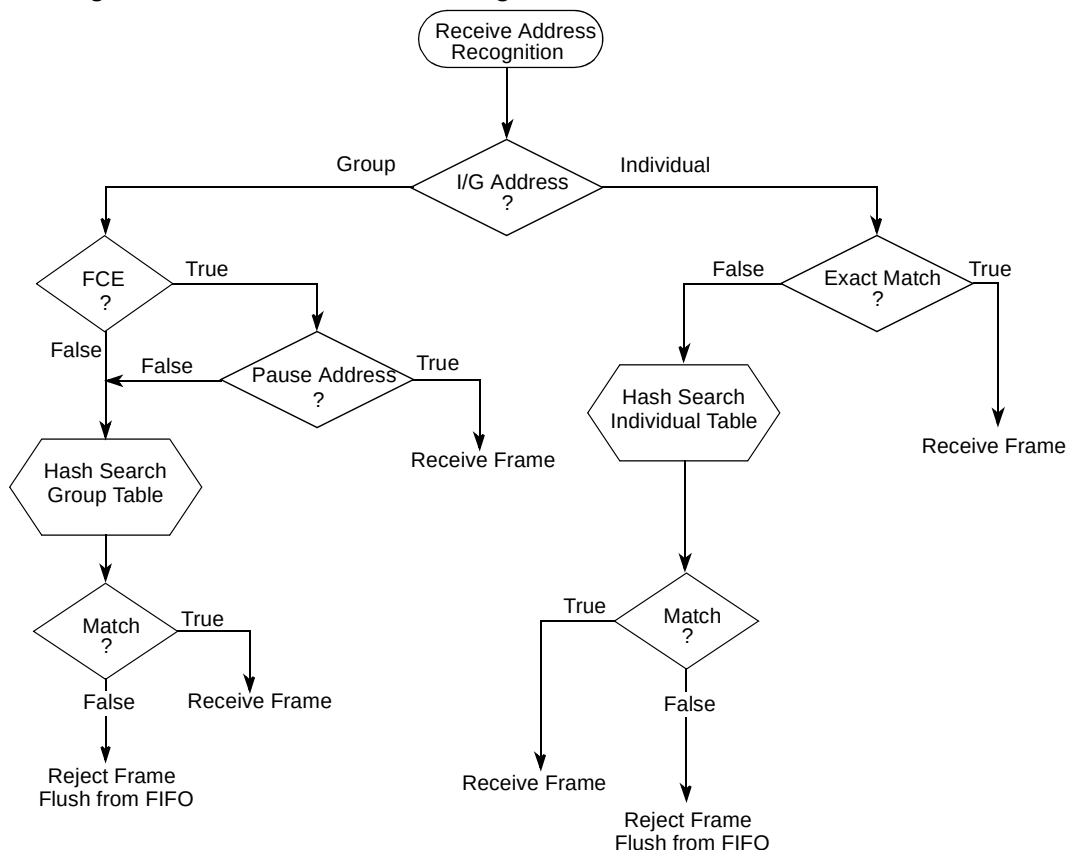


Figure 936. Ethernet address recognition—microcode decisions**Notes:**

FCE - field in RCR register (flow control enable)

I/G - Individual/Group bit in destination address (lsb in first byte received in MAC frame)

50.5.12 Hash algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits stored in GAUR, GALR (group address hash match), or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The msb of the CRC result selects GAUR (msb = 1) or GALR (msb = 0). The five least significant bits of the hash result select the bit within the selected register. If the CRC generator selects a bit set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

Equation 38
$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

[Table 967](#) contains example destination addresses and corresponding hash values.

Table 967. Destination address to 6-bit hash

48-bit DA	6-bit hash (in hex)	Hash decimal value
65FF_FFFF_FFFF	0h	0
55FF_FFFF_FFFF	1h	1
15FF_FFFF_FFFF	2h	2
35FF_FFFF_FFFF	3h	3
B5FF_FFFF_FFFF	4h	4
95FF_FFFF_FFFF	5h	5
D5FF_FFFF_FFFF	6h	6
F5FF_FFFF_FFFF	7h	7
DBFF_FFFF_FFFF	8h	8
FBFF_FFFF_FFFF	9h	9
BBFF_FFFF_FFFF	Ah	10
8BFF_FFFF_FFFF	Bh	11
0BFF_FFFF_FFFF	Ch	12
3BFF_FFFF_FFFF	Dh	13
7BFF_FFFF_FFFF	Eh	14
5BFF_FFFF_FFFF	Fh	15
27FF_FFFF_FFFF	10h	16
07FF_FFFF_FFFF	11h	17
57FF_FFFF_FFFF	12h	18
77FF_FFFF_FFFF	13h	19
F7FF_FFFF_FFFF	14h	20
C7FF_FFFF_FFFF	15h	21
97FF_FFFF_FFFF	16h	22
A7FF_FFFF_FFFF	17h	23
99FF_FFFF_FFFF	18h	24
B9FF_FFFF_FFFF	19h	25
F9FF_FFFF_FFFF	1Ah	26
C9FF_FFFF_FFFF	1Bh	27
59FF_FFFF_FFFF	1Ch	28
79FF_FFFF_FFFF	1Dh	29
29FF_FFFF_FFFF	1Eh	30
19FF_FFFF_FFFF	1Fh	31
D1FF_FFFF_FFFF	20h	32

Table 967. Destination address to 6-bit hash (continued)

48-bit DA	6-bit hash (in hex)	Hash decimal value
F1FF_FFFF_FFFF	21h	33
B1FF_FFFF_FFFF	22h	34
91FF_FFFF_FFFF	23h	35
11FF_FFFF_FFFF	24h	36
31FF_FFFF_FFFF	25h	37
71FF_FFFF_FFFF	26h	38
51FF_FFFF_FFFF	27h	39
7FFF_FFFF_FFFF	28h	40
4FFF_FFFF_FFFF	29h	41
1FFF_FFFF_FFFF	2Ah	42
3FFF_FFFF_FFFF	2Bh	43
BFFF_FFFF_FFFF	2Ch	44
9FFF_FFFF_FFFF	2Dh	45
DFFF_FFFF_FFFF	2Eh	46
EFFF_FFFF_FFFF	2Fh	47
93FF_FFFF_FFFF	30h	48
B3FF_FFFF_FFFF	31h	49
F3FF_FFFF_FFFF	32h	50
D3FF_FFFF_FFFF	33h	51
53FF_FFFF_FFFF	34h	52
73FF_FFFF_FFFF	35h	53
23FF_FFFF_FFFF	36h	54
13FF_FFFF_FFFF	37h	55
3DFF_FFFF_FFFF	38h	56
0DFF_FFFF_FFFF	39h	57
5DFF_FFFF_FFFF	3Ah	58
7DFF_FFFF_FFFF	3Bh	59
FDFF_FFFF_FFFF	3Ch	60
DDFF_FFFF_FFFF	3Dh	61
9DFF_FFFF_FFFF	3Eh	62
BDFF_FFFF_FFFF	3Fh	63

50.5.13 Full duplex flow control

Full-duplex flow control allows you to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable PAUSE frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] set) with flow control (RCR[FCE] set). The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in [Table 968](#). In addition, the receive status associated with the frame should indicate that the frame is valid.

Table 968. PAUSE frame field specification

48-bit destination address	0180_C200_0001h or Physical Address
48-bit source address	Any
16-bit type	8808h
16-bit opcode	0001h
16-bit PAUSE duration	0000h – FFFFh

The receiver and microcontroller modules perform PAUSE frame detection. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is set by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. The pause timer uses the transmit backoff timer hardware for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is cleared allowing MAC data frame transmission to resume. The receive flow control pause status bit (TCR[RFC_PAUSE]) is set while the transmitter pauses due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and you must set flow control pause (TCR[TFC_PAUSE]). After TCR[TFC_PAUSE] is set, the transmitter sets TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts and the pause frame is transmitted. TCR[TFC_PAUSE,GTS] are then cleared internally.

You must specify the desired pause duration in the OPD register.

When the transmitter pauses due to receiver/microcontroller pause frame detection, TCR[TFC_PAUSE] may remain set and cause the transmission of a single pause frame. In this case, the EIR[GRA] interrupt is not asserted.

50.5.14 Inter-Packet Gap (IPG) time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it is ignored and a collision occurs.

The receiver accepts back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times, the receiver may discard the following frame.

50.5.15 Collision managing

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the

sequence, a JAM pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit times (one slot time), the retry process is initiated. The transmitter waits a random number of slot times. If a collision occurs after 512 bit times, then no retransmission is performed and the end of frame buffer is closed with a Late Collision (LC) error indication.

50.5.16 MII internal and external loopback

Internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Internal and external loopback are configured using combinations of the RCR[LOOP, DRT] and TCR[FDEN] bits.

Set FDEN for internal and external loopback.

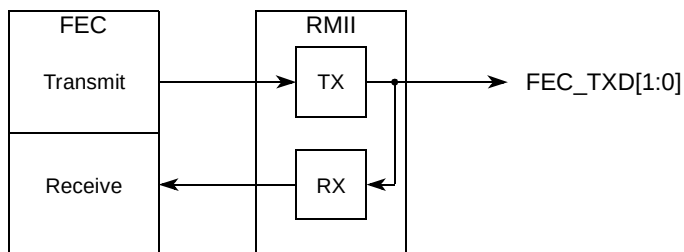
For internal loopback, set RCR[LOOP] and clear RCR[DRT]. FEC_TXEN and FEC_TXER do not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the transmit and receive blocks use the internal bus clock instead of the clocks from the external transceiver. This causes an increase in the required system bus bandwidth for transmit and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underruns and receive FIFO overflows.

For external loopback, clear RCR[LOOP] and RCR[DRT], and configure the external transceiver for loopback.

50.5.17 RMII loopback

The FEC also supports RMII loopback. In this mode, transmit data is sent to the RMII receive logic and out the FEC_TXD[1:0] pins. To enable RMII loopback mode, set RCR[RMII_MODE, RMII_LOOP] and TCR[FDEN] and clear RCR[RMII_ECHO, LOOP].

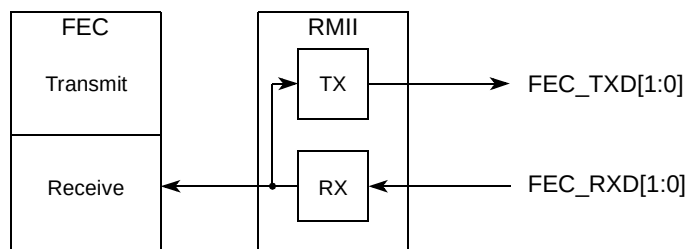
Figure 937. RMII loopback mode



50.5.18 RMII echo

The Ethernet controller also supports RMII echo mode, which transmits data on FEC_TXD[1:0] as it is received on FEC_RXD[1:0]. To enable RMII echo mode, set RCR[RMII_MODE, RMII_ECHO] bits and clear RCR[RMII_LOOP, LOOP].

Figure 938. RMII echo mode



50.5.19 Ethernet error-managing procedure

The Ethernet controller reports frame reception and transmission error conditions using the MIB block counters, the FEC RxBDs, and the EIR register.

50.5.19.1 Transmission errors

50.5.19.1.1 Transmitter underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed, and EIR[UN] is set. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame. The UN interrupt is asserted if enabled in the EIMR register.

50.5.19.1.2 Retransmission attempts limit expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[RL] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The RL interrupt is asserted if enabled in the EIMR register.

50.5.19.1.3 Late collision

When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[LC] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The LC interrupt is asserted if enabled in the EIMR register.

50.5.19.1.4 Heartbeat

Some transceivers have a self-test feature called heartbeat or signal quality error. To signify a good self-test, the transceiver indicates a collision to the FEC within four microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver continues to function properly. This is the heartbeat condition.

If TCR[HBC] is set and the heartbeat condition is not detected by the **FEC** after a frame transmission, a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets EIR[HB], and generates the HBERR interrupt if it is enabled.

50.5.19.2 Reception errors

50.5.19.2.1 Overrun error

If the receive block has data to put into the receive FIFO and the receive FIFO is full, FEC sets RxB[OV]. All subsequent data in the frame is discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. The driver must discard this frame.

50.5.19.2.2 Non-octet error (dribbling bits)

The Ethernet controller manages up to seven dribbling bits when the receive frame terminates past a non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, the frame non-octet aligned (NO) error is reported in the RxB. If there is no CRC error, no error is reported.

50.5.19.2.3 CRC error

When a CRC error occurs with no dribble bits, FEC closes the buffer and sets RxB[CR]. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

50.5.19.2.4 Frame length violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and RxB[LG] is set. The frame is not truncated unless the frame length exceeds 2047 bytes.

50.5.19.2.5 Truncation

When the receive frame length exceeds 2047 bytes, frame is truncated and RxB[TR] is set.

51 FlexRay Communication Controller (FlexRay)

51.1 Introduction

51.1.1 Reference

The following documents are referenced in this chapter. They may be consulted for further information on the FlexRay Communication Controller's operation.

- FlexRay Communications System Protocol Specification, Version 2.1 Rev A^(q)
- FlexRay Communications System Electrical Physical Layer Specification, Version 3.0

51.1.2 Glossary

[Table 969](#) provides a list of terms used in this chapter.

Table 969. List of terms

Term	Definition
BCU	Buffer Control Unit. Handles message buffer access.
BMIF	Bus Master Interface. Provides master access to FlexRay memory area.
CC	Communication Controller
CDC	Clock Domain Crosser
CHI	Controller Host Interface
Cycle length in μT	The actual length of a cycle in μT for the ideal controller (+/- 0 ppm)
EBI	External Bus Interface
FlexRay memory area	Memory area for storing the physical message buffer payload data, frame header, frame and slot status, and synchronization frame-related tables.
System memory	Memory that contains the FlexRay memory area.
System bus	Bus that connects the controller and system memory
FSS	Frame Start Sequence
HIF	Host Interface. Provides host access to controller.
Host	The FlexRay CC host MCU
LUT	Look Up Table. Stores message buffer header index value.
LRAM	Look Up Table RAM. Module internal memory for storing message buffer configuration data and data field offsets for individual message buffers and receive shadow buffers.
MB	Message Buffer
MBIDX	Message Buffer Index. Position of a header field entry within the header area. If the header area is accessed as an array, this is the same as the array index of the entry.

q. The FlexRay Specifications have been developed for automotive applications. The FlexRay Specifications have been neither developed nor tested for non-automotive applications.

Table 969. List of terms (continued)

Term	Definition
MNum	Message Buffer Number. Position of message buffer configuration registers within the register map. For example, Message Buffer Number 5 corresponds to the MBCCS5 register.
MCU	Microcontroller Unit
μ T	Microtick
MT	Macrotick
MTS	Media Access Test Symbol
NIT	Network Idle Time
PE	Protocol Engine
POC	Protocol Operation Control. Each state of the POC is denoted by <i>POC:state</i>
Rx	Reception
SEQ	Sequencer Engine
SEDED	Single-bit error correction, double-bit error detection
TCU	Time Control Unit
Tx	Transmission
sync frame	null frame or message frame with <i>Sync Frame Indicator</i> set to 1
startup frame	null frame or message frame with both <i>Sync Frame Indicator</i> and <i>Startup Frame Indicator</i> set to 1
normal frame	null frame or message frame with both <i>Sync Frame Indicator</i> and <i>Startup Frame Indicator</i> set to 0
null frame	frame with <i>Null Frame Indicator</i> set to 0
message frame	frame with <i>Null Frame Indicator</i> set to 1

51.1.3 Color coding

Throughout this chapter, some items are displayed in an italicized, colored font.

FlexRay protocol parameters, constants, and variables are highlighted with *blue italics*. An example is the parameter *gdActionPointOffset*.

FlexRay protocol states are highlighted in *green italics*. An example is the state *POC:normal active*.

51.1.4 Overview

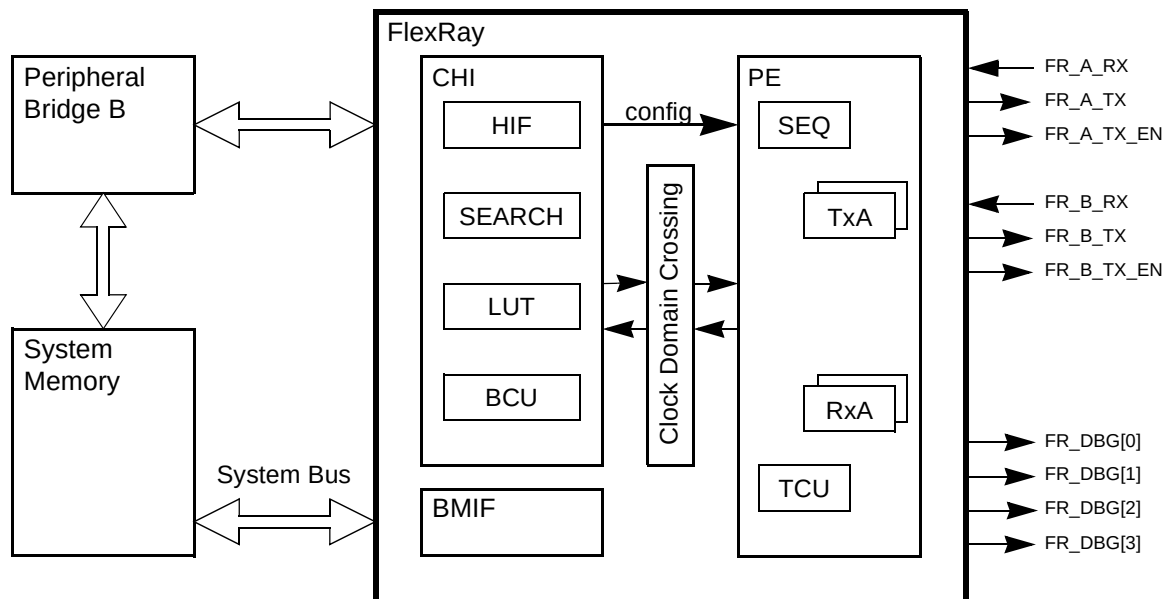
The communication controller (CC) is a FlexRay communication controller that implements the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

The CC has three main components:

- Controller host interface (CHI)
- Protocol engine (PE)
- Clock domain crossing unit (CDC)

A block diagram of the CC with its surrounding modules is given in [Figure 939](#).

Figure 939. FlexRay block diagram



The protocol engine has two transmitter units, TxA and TxB, and two receiver units, RxA and RxB, for sending and receiving frames through the two FlexRay channels. The time control unit (TCU) is responsible for maintaining global clock synchronization to the FlexRay network. The overall activity of the protocol engine (PE) is controlled by the sequencer engine (SEQ).

The controller host interface provides host access to the module's configuration, control, and status registers, as well as to the message buffer configuration, control, and status registers. The message buffers themselves, which contain the frame header and payload data received or to be transmitted, as well as the slot status information, are stored in the FlexRay memory area.

The clock domain crossing unit implements signal crossing from the controller host interface (CHI) clock domain to the PE clock domain and vice versa, to allow for asynchronous PE and CHI clock domains.

The CC stores the frame header and payload data of frames received or of frames to be transmitted in the FlexRay memory area. The application accesses the FlexRay memory area to retrieve and provide the frames to be processed by the CC. In addition to the frame header and payload data, the CC stores the synchronization frame related tables in the FlexRay memory area for application processing.

The FlexRay memory area is located in the system memory of the MCU. The CC has access to the FlexRay memory area via its bus master interface (BMIF). The host provides the start address of the FlexRay memory area within the system memory by programming the [System Memory Base Address Register \(FR_SYMBADR\)](#). All FlexRay memory area related offsets are stored in offset registers. The physical address pointer into the FlexRay memory area is calculated using the offset values the FlexRay memory base address.

Note: The CC does not provide a memory protection scheme for the FlexRay memory area.

51.1.5 Features

The CC provides the following features:

- FlexRay Communications System Protocol Specification, Version 2.1 Rev A-compliant protocol implementation
- FlexRay Communications System Electrical Physical Layer Specification, Version 3.0-compliant bus driver interface
- Single channel support
 - FlexRay Port A can be configured to be connected either to physical FlexRay channel A or physical FlexRay channel B.
- FlexRay bus data rates of 10 Mbit/s, 8 Mbit/s, 5 Mbit/s, and 2.5 Mbit/s supported
- 128 configurable message buffers with
 - Individual frame ID filtering
 - Individual channel ID filtering
 - Individual cycle counter filtering
- Message buffer header, status, and payload data stored in dedicated FlexRay memory area
 - Allows for flexible and efficient message buffer implementation
 - Ensures consistent data access by means of a buffer-locking scheme
 - Application can lock multiple buffers at the same time
- Size of message buffer payload data section configurable from 0–254 bytes
- Two independent message buffer segments with configurable size of payload data section
 - Each segment can contain message buffers assigned to the static segment and message buffers assigned to the dynamic segment at the same time
- Zero padding for transmit message buffers in static segment
 - Applied when the frame payload length exceeds the size of the message buffer data section
- Transmit message buffers configurable with state/event semantics
- Message buffers can be configured as
 - Receive message buffer
 - Transmit message buffer
- Individual message buffers can be reconfigured
 - Individual message buffers can be disabled safely
 - Disabled message buffers can be reconfigured
- Two independent receive FIFOs
 - One receive FIFO per channel
 - Up to 255 entries for each FIFO
 - Global frame ID filtering, based on both value/mask filters and range filters
 - Global channel ID filtering
 - Global message ID filtering for the dynamic segment
- Four configurable slot error counters
- Four dedicated slot status indicators
 - Used to observe slots without using receive message buffers

- Measured value indicators for clock synchronization
 - Internal synchronization frame ID and synchronization frame measurement tables can be copied into the FlexRay memory area
- Fractional macroticks are supported for clock correction
- Maskable interrupt sources provided via individual and combined interrupt lines
- One absolute timer
- One timer that can be configured as absolute or relative
- SECEDED for protocol engine data RAM
- SEDDED for CHI lookup table RAM

51.1.6 Modes of operation

This section describes the basic operational power modes of the CC.

51.1.6.1 Disabled mode

The CC enters Disabled mode during hard reset or when the host clears the 'MEN' field in the Module Configuration Register (FR_MCR). The host can clear this field by writing '0' and only when the module is in *POC:default config* mode. The Disabled mode can be checked by reading the module enable field, MEN, in the *Module Configuration Register (FR_MCR)*.

No communication is performed on the FlexRay bus.

All registers with the write access conditions *Any Time* and *Disabled mode* can be accessed for writing as stated in *Section 51.5, "Memory map and register description"*.

The application configures the CC by accessing the configuration fields in the *Module Configuration Register (FR_MCR)* as described in *Section 51.7.2.1, "Module initialization"*.

51.1.6.1.1 Leave Disabled mode

The CC leaves Disabled mode and enters Normal mode, when the application writes 1 to *FR_MCR[MEN]*.

Note: Once the CC is enabled, it cannot be disabled later.

51.1.6.2 Normal mode

In this mode, the CC is fully functional. The CC indicates that it is in Normal mode by asserting *FR_MCR[EN]*.

51.1.6.2.1 Enter Normal mode

This mode is entered when the application requests that the CC leave Disabled mode. If Normal mode is entered after leaving Disabled mode, the application has to perform the protocol initialization described in *Section 51.7.2.2, "Protocol initialization"* to achieve full FlexRay functionality.

Depending on the values of the SCM, CHA, and CHB fields in the *Module Configuration Register (FR_MCR)*, the corresponding FlexRay bus driver ports are enabled and driven.

51.1.6.3 System memory test

51.1.6.3.1 Bus Master Interface Functional Test for AMBA/AHB Bus

This section describes the test access to test the AMBA/AHB Bus Master Interface.

51.1.6.3.1.1 AMBA Read Access

This describes the sequence to initiate a amba read access

1. write word offset of address to read from into FR_PEDRAR[SYSMEMADDR]
2. and set the read/write indicator FR_PEDRAR[RW] to 1

The module clears the FR_PEDRAR[DAD] bit while access is running

The BMIF initiates a read from 64 bit-aligned byte address:

readaddr = ({SYSMEMADDR,0} + SMBA) & 64'hFFFFFFF8

The module sets the FR_PEDRAR[DAD] bit when access has finished

3. the data read and corresponds to {SYSMEMADDR,0} + SMBA are provided in the FR_PEDRDR register

51.1.6.3.1.2 AMBA Write Access (Prepare)

This describes how to provide data for the AMBA write access.

1. write 16-bit data to be written into FR_PEDRDR register
2. write 16-bit offset of address to be written to into FR_PEDRAR[SYSMEMADDR]
3. and set the read/write indicator FR_PEDRAR[RW] to 0

The module clears the FR_PEDRAR[DAD] bit.

The BMIF buffers this write request,

4. the module sets the FR_PEDRAR[DAD] bit when data are buffered

51.1.6.3.1.3 AMBA Write Access (Initiate)

The BMIF initiates the AMBA write access if any of sequences listed above was executed.
writeaddr = ({SYSMEMADDR,0} + SMBA) & 64'hFFFFFFF8.

Note: the read access to force the buffer write is not necessary in all cases, but suggested to have similar test sequences.

51.1.6.3.1.4 16 bit write initiate

A 16-bit write can be initiated as follows

- write to byte 0 and byte 1:
initiate read from any address, to empty the buffer
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 00
initiate read from any address, to force buffer write
- write to byte 2 and byte 3:
initiate read from any address, to empty the buffer
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 01
initiate read from any address, to force buffer write
- write to byte 4 and byte 5:
initiate read from any address, to empty the buffer
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 10
initiate read from any address, to force buffer write
- write to byte 6 and byte 7:
initiate read from any address, to empty the buffer
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 11
initiate read from any address, to force buffer write

51.1.6.3.1.5 32 bit write

- write to byte 0,1,2,3:
initiate read from any address, to empty the buffer
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 00
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 01
initiate read from any address, to force buffer write
- write to byte 4,5,6,7:
initiate read from any address, to empty the buffer
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 10
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 11
initiate read from any address, to force buffer write

51.1.6.3.1.6 64 bit write

- write to byte 0,1,2,3,4,5,6,7:
initiate read from any address, to empty the buffer
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 00
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 01
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 10
write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 11
initiate read from any address, to force buffer write

51.2 External Signal Description

This section lists and describes the CC signals connected to external pins. These signals are summarized in [Table 970](#) and described in detail in [Section 51.2.1, “Detailed signal description”](#).

Note: The off-chip FR_A_RX, FR_A_TX, and FR_A_TX_EN signals are available on each package option. The availability of the other off-chip signals depends on the package option and is chip-specific.

Table 970. External Signal Properties

Name	Direction	Active	Reset	Function
FR_A_RX	Input	—	—	Receive Data Channel A
FR_A_TX	Output	—	1	Transmit Data Channel A
FR_A_TX_EN	Output	Low	1	Transmit Enable Channel A
FR_B_RX	Input	—	—	Receive Data Channel B
FR_B_TX	Output	—	1	Transmit Data Channel B
FR_B_TX_EN	Output	Low	1	Transmit Enable Channel B
FR_DBG[0]	Output	—	0	Debug Strobe Signal 0
FR_DBG[1]	Output	—	0	Debug Strobe Signal 1
FR_DBG[2]	Output	—	0	Debug Strobe Signal 2
FR_DBG[3]	Output	—	0	Debug Strobe Signal 3

51.2.1 Detailed signal description

This section provides a detailed description of the CC signals connected to external pins.

51.2.1.1 FR_A_RX—Receive Data Channel A

The FR_A_RX signal carries the receive data for channel A from the corresponding FlexRay bus driver.

51.2.1.2 FR_A_TX—Transmit Data Channel A

The FR_A_TX signal carries the transmit data for channel A to the corresponding FlexRay bus driver.

51.2.1.3 FR_A_TX_EN—Transmit Enable Channel A

The FR_A_TX_EN signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel A.

51.2.1.4 FR_B_RX—Receive Data Channel B

The FR_B_RX signal carries the receive data for channel B from the corresponding FlexRay bus driver.

51.2.1.5 FR_B_TX—Transmit Data Channel B

The FR_B_TX signal carries the transmit data for channel B to the corresponding FlexRay bus driver.

51.2.1.6 FR_B_TX_EN—Transmit Enable Channel B

The FR_B_TX_EN signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel B.

51.2.1.7 FR_DBG[3], FR_DBG[2], FR_DBG[1], FR_DBG[0]—Strobe Signals

These signals provide the selected debug strobe signals. For details on the debug strobe signal selection, refer to [Section 51.6.16, “Strobe signal support”](#).

51.3 Controller Host Interface clocking

The clock for the CHI is derived from the system bus clock and has the same phase and frequency as the system bus clock. There are two constraints for the minimum CHI clock frequency.

- The first constraint corresponds to the number of utilized message buffers and is specified in [Section 51.7.5, “Number of usable message buffers”](#).
- The second constraint corresponds to the value of the TIMEOUT field in the [System Memory Access Time-Out Register \(FR_SYMATOR\)](#) and is specified in [Section 51.7.1.1, “Configure System Memory Access Timeout Register \(FR_SYMATOR\)”](#).

51.4 Protocol engine clocking

The clock for the protocol engine can be generated by two sources. The first source is the internal crystal oscillator and the second source is an internal PLL. The clock source to be used is selected by the clock source select field, CLKSEL, in the [Module Configuration Register \(FR_MCR\)](#).

Note: See the device clocking details for the exact clock sources used.

51.4.1 Oscillator clocking

If the protocol engine is clocked by the internal crystal oscillator, a 40 MHz crystal or CMOS-compatible clock must be connected to the oscillator pins. The crystal or clock must fulfill the requirements given by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

51.4.2 PLL clocking

If the protocol engine is clocked by the internal PLL, the PLL clock should be divided down appropriately to generate the 40 MHz and 180 degree phase-shifted 40 MHz clocks and provide them to the PE. For more details, see the clocking chapter of the device reference manual.

51.5 Memory map and register description

The CC occupies 8 KB (8192 bytes) of address space starting at the CC base address defined by the memory map of the MCU.

51.5.1 Memory map

The complete memory map of the CC is shown in [Table 971](#). The addresses presented here are the offsets relative to the CC base address, which is defined by the MCU address map.

Table 971. FlexRay memory map

Address Offset	Register Name	Access	Reset value	Section/ Page
Module Configuration and Control				
0x0000	Module Version Register (FR_MVR)	R	0xA468	51.5.2.3/1897
0x0002	Module Configuration Register (FR_MCR)	R/W	0x0000	51.5.2.4/1897
0x0004	System Memory Base Address High Register (FR_SYMBADHR)	R/W	0x0000	51.5.2.5/1900
0x0006	System Memory Base Address Low Register (FR_SYMBADLR)	R/W	0x0000	51.5.2.5/1900
0x0008	Strobe Signal Control Register (FR_STBSCR)	R/W	0x0000	51.5.2.6/1900
0x000A	Reserved	R	0x0000	—
0x000C	Message Buffer Data Size Register (FR_MBDSR)	R/W	0x0000	51.5.2.7/1902
0x000E	Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)	R/W	0x7F7F	51.5.2.8/1902
PE Access Registers				
0x0010	PE DRAM Access Register (FR_PEDRAR)	R/W	0x0000	51.5.2.9/1903
0x0012	PE DRAM Data Register (FR_PEDRDR)	R/W	0x0000	51.5.2.10/1904
Interrupt and Error Handling				
0x0014	Protocol Operation Control Register (FR_POCR)	R/W	0x0000	51.5.2.11/1905
0x0016	Global Interrupt Flag and Enable Register (FR_GIFER)	R/W	0x0000	51.5.2.12/1906
0x0018	Protocol Interrupt Flag Register 0 (FR_PIFR0)	R/W	0x0000	51.5.2.13/1909
0x001A	Protocol Interrupt Flag Register 1 (FR_PIFR1)	R/W	0x0000	51.5.2.14/1911
0x001C	Protocol Interrupt Enable Register 0 (FR_PIER0)	R/W	0x0000	51.5.2.15/1912
0x001E	Protocol Interrupt Enable Register 1 (FR_PIER1)	R/W	0x0000	51.5.2.16/1914
0x0020	CHI Error Flag Register (FR_CHIERFR)	R/W	0x0000	51.5.2.17/1915
0x0022	Message Buffer Interrupt Vector Register (FR_MBIVEC)	R	0x0000	51.5.2.18/1918
0x0024	Channel A Status Error Counter Register (FR_CASERCR)	R	0x0000	51.5.2.19/1918
0x0026	Channel B Status Error Counter Register (FR_CBSERCR)	R	0x0000	51.5.2.20/1919
Protocol Status				
0x0028	Protocol Status Register 0 (FR_PSR0)	R	0x0000	51.5.2.21/1919
0x002A	Protocol Status Register 1 (FR_PSR1)	R	0x0000	51.5.2.22/1921
0x002C	Protocol Status Register 2 (FR_PSR2)	R	0x0000	51.5.2.23/1922
0x002E	Protocol Status Register 3 (FR_PSR3)	R/W	0x0000	51.5.2.24/1924
0x0030	Macrotick Counter Register (FR_MTCTR)	R	0x0000	51.5.2.25/1925
0x0032	Cycle Counter Register (FR_CYCTR)	R	0x0000	51.5.2.26/1926
0x0034	Slot Counter Channel A Register (FR_SLCTAR)	R	0x0000	51.5.2.27/1926

Table 971. FlexRay memory map (continued)

Address Offset	Register Name	Access	Reset value	Section/ Page
0x0036	Slot Counter Channel B Register (FR_SLCTCTBR)	R	0x0000	51.5.2.28/1927
0x0038	Rate Correction Value Register (FR_RTCORVR)	R	0x0000	51.5.2.29/1927
0x003A	Offset Correction Value Register (FR_OFCORVR)	R	0x0000	51.5.2.30/1928
0x003C	Combined Interrupt Flag Register (FR_CIFR)	R	0x0000	51.5.2.31/1928
0x003E	System Memory Access Time-Out Register (FR_SYMATOR)	R/W	0x0006	51.5.2.32/1929
Sync Frame Counter and Tables				
0x0040	Sync Frame Counter Register (FR_SFCNTR)	R	0x0000	51.5.2.33/1930
0x0042	Sync Frame Table Offset Register (FR_SFTOR)	R/W	0x0000	51.5.2.34/1931
0x0044	Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)	R/W	0x0000	51.5.2.35/1931
Sync Frame Filter				
0x0046	Sync Frame ID Rejection Filter Register (FR_SFIDRFR)	R/W	0x0000	51.5.2.36/1933
0x0048	Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)	R/W	0x0000	51.5.2.37/1933
0x004A	Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)	R/W		51.5.2.38/1934
Network Management Vector				
0x004C	Network Management Vector Register 0 (FR_NMVR0)	R	0x0000	51.5.2.39/1934
0x004E	Network Management Vector Register 1 (FR_NMVR1)	R	0x0000	
0x0050	Network Management Vector Register 2 (FR_NMVR2)	R	0x0000	
0x0052	Network Management Vector Register 3 (FR_NMVR3)	R	0x0000	
0x0054	Network Management Vector Register 4 (FR_NMVR4)	R	0x0000	
0x0056	Network Management Vector Register 5 (FR_NMVR5)	R	0x0000	51.5.2.40/1935
0x0058	Network Management Vector Length Register (FR_NMVLR)	R/W	0x0000	
Timer Configuration				
0x005A	Timer Configuration and Control Register (FR_TICCR)	R/W	0x0000	51.5.2.41/1936
0x005C	Timer 1 Cycle Set Register (FR_TI1CYSR)	R/W	0x0000	51.5.2.42/1937
0x005E	Timer 1 Macrotick Offset Register (FR_TI1MTOR)	R/W	0x0000	51.5.2.43/1937
0x0060	Timer 2 Configuration Register 0 (FR_TI2CR0)	R/W	0x0000	51.5.2.44/1938
0x0062	Timer 2 Configuration Register 1 (FR_TI2CR1)	R/W	0x0000	51.5.2.45/1938
Slot Status Configuration				
0x0064	Slot Status Selection Register (FR_SSSR)	R/W	0x0000	51.5.2.46/1939

Table 971. FlexRay memory map (continued)

Address Offset	Register Name	Access	Reset value	Section/ Page
0x0066	Slot Status Counter Condition Register (FR_SSCCR)	R/W	0x0000	51.5.2.47/1940
Slot Status				
0x0068	Slot Status Register 0 (FR_SSR0)	R	0x0000	51.5.2.48/1942
0x006A	Slot Status Register 1 (FR_SSR1)	R	0x0000	
0x006C	Slot Status Register 2 (FR_SSR2)	R	0x0000	
0x006E	Slot Status Register 3 (FR_SSR3)	R	0x0000	
0x0070	Slot Status Register 4 (FR_SSR4)	R	0x0000	
0x0072	Slot Status Register 5 (FR_SSR5)	R	0x0000	
0x0074	Slot Status Register 6 (FR_SSR6)	R	0x0000	
0x0076	Slot Status Register 7 (FR_SSR7)	R	0x0000	
0x0078	Slot Status Counter Register 0 (FR_SSCR0)	R	0x0000	51.5.2.49/1944
0x007A	Slot Status Counter Register 1 (FR_SSCR1)	R	0x0000	
0x007C	Slot Status Counter Register 2 (FR_SSCR2)	R	0x0000	
0x007E	Slot Status Counter Register 3 (FR_SSCR3)	R	0x0000	
MTS Generation				
0x0080	MTS A Configuration Register (FR_MTSACFR)	R/W	0x0000	51.5.2.50/1944
0x0082	MTS B Configuration Register (FR_MTSBCFR)	R/W	0x0000	51.5.2.51/1945
Shadow Buffer Configuration				
0x0084	Receive Shadow Buffer Index Register (FR_RSBIR)	R/W	0x0000	51.5.2.52/1945
Receive FIFO—Configuration				
0x0086	Receive FIFO Watermark and Selection Register (FR_RFWMSR)	R/W	0x0000	51.5.2.56/1948
0x0088	Receive FIFO Start Index Register (FR_RFSIR)	R/W	0x0000	51.5.2.57/1949
0x008A	Receive FIFO Depth and Size Register (FR_RFDSR)	R/W	0x0000	51.5.2.58/1949
Receive FIFO—Control				
0x008C	Receive FIFO A Read Index Register (FR_RFARIR)	R	0x0000	51.5.2.59/1950
0x008E	Receive FIFO B Read Index Register (FR_RFBIRIR)	R	0x0000	51.5.2.60/1950
Receive FIFO—Filter				
0x0090	Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)	R/W	0x0000	51.5.2.62/1951
0x0092	Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)	R/W	0x0000	51.5.2.63/1952
0x0094	Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)	R/W	0x0000	51.5.2.64/1952

Table 971. FlexRay memory map (continued)

Address Offset	Register Name	Access	Reset value	Section/ Page
0x0096	Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)	R/W	0x0000	51.5.2.65/1953
0x0098	Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)	R/W	0x0000	51.5.2.66/1953
0x009A	Receive FIFO Range Filter Control Register (FR_RFRFCTR)	R/W	0x0000	51.5.2.67/1954
Dynamic Segment Status				
0x009C	Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)	R	0x0000	51.5.2.68/1955
0x009E	Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)	R	0x0000	51.5.2.69/1955
Protocol Configuration				
0x00A0 ... 0x00DC	Protocol Configuration Register 0 (FR_PCR0) ... Protocol Configuration Register 30 (FR_PCR30)	R/W	0x0000	51.5.2.70.30/1965 ... 51.5.2.70.31/1966
0x00DE	Stop Watch Count High Register (FR_STPWHR)	RR	0x0000	51.5.2.86/1979
0x00E0	Stop Watch Count Low Register (FR_STPWLR)	RR	0x0000	51.5.2.86/1979
Event Output and StopWatch Configuration				
0x00E2	Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)	R/W	0x0000	51.5.2.85/1979
0x00E4	Reserved	R	0x0000	—
Receive FIFO—Configuration				
0x00E6	Receive FIFO Start Data Offset Register (FR_RFSDOR)	R/W	0x0000	51.5.2.53/1946
0x00E8	Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)	R/W	0x0000	51.5.2.54/1947
0x00EA	Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)	R/W	0x0000	51.5.2.55/1947
0x00EC	Receive FIFO Periodic Timer Register (FR_RFPTR)	R/W	0x0000	51.5.2.56/1948
Receive FIFO—Control				
0x00EE	Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)	R/W	0x0000	51.5.2.61/1951

Table 971. FlexRay memory map (continued)

Address Offset	Register Name	Access	Reset value	Section/ Page
ECC Registers				
0x00F0	ECC Error Interrupt Flag and Enable Register (FR_EEIFER)	R/W	0x0000	51.5.2.71/1966
0x00F2	ECC Error Report and Injection Control Register (FR_EERICR)	R/W	0x0000	51.5.2.72/1969
0x00F4	ECC Error Report Address Register (FR_EERAR)	R	0x7000	51.5.2.73/1970
0x00F6	ECC Error Report Data Register (FR_EERDR)	R	0x0000	51.5.2.74/1971
0x00F8	ECC Error Report Code Register (FR_EERCR)	R	0x0000	51.5.2.75/1972
0x00FA	ECC Error Injection Address Register (FR_EEIAR)	R/W	0x0000	51.5.2.76/1972
0x00FC	ECC Error Injection Data Register (FR_EEIDR)	R/W	0x0000	51.5.2.77/1973
0x00FE	ECC Error Injection Code Register (FR_EEICR)	R/W	0x0000	51.5.2.78/1974
0x0100 ... 0x07FE	Not Implemented	R	—	—
Message Buffers Configuration, Control, Status				
0x0800	Message Buffer Configuration, Control, Status Register 0 (FR_MBCCSR0)	R/W	0x0000	51.5.2.79/1974

Table 971. FlexRay memory map (continued)

Address Offset	Register Name	Access	Reset value	Section/ Page
0x0802	Message Buffer Cycle Counter Filter Register 0 (FR_MBCCFR0)	R/W	—	51.5.2.80/1976
0x0804	Message Buffer Frame ID Register 0 (FR_MBFIDR0)	R/W	0x0---	51.5.2.81/1977
0x0806	Message Buffer Index Register 0 (FR_MBIDXR0)	R/W	0x00--	51.5.2.82/1977
...	...	...		
0x0BF8	Message Buffer Configuration, Control, Status Register 127 (FR_MBCCSR127)	R/W	0x0000	51.5.2.79/1974
0x0BFA	Message Buffer Cycle Counter Filter Register 127 (FR_MBCCFR127)	R/W	—	51.5.2.80/1976
0x0BFC	Message Buffer Frame ID Register 127 (FR_MBFIDR127)	R/W	0x0---	51.5.2.81/1977
0x0BFE	Message Buffer Index Register 127 (FR_MBIDXR127)	R/W	0x00--	51.5.2.82/1977
0x0C00 ... 0x0FFF	Not Implemented ¹	R	—	—
0x1000 ... 0x1106	Message Buffer Data Field Offset Register 0 (FR_MBDOR0) ... Message Buffer Data Field Offset Register 131 (FR_MBDOR131)	R/W	—	51.5.2.83/1978
0x1108 ... 0x1112	LRAM ECC Error Test Register 0 (FR_LEETR0) ... LRAM ECC Error Test Register 5 (FR_LEETR5)	R/W	—	51.5.2.84/1978
0x1114 ... 0x1FFE	Not Implemented ¹	R		—

51.5.2 Register descriptions

This section provides detailed descriptions of all registers in ascending address order, presented as 16-bit wide entities

51.5.2.1 Register Reset

Table 972. Register access conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
R*	Reserved bit or field, will not be changed. Application must not write any value different from the reset value.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A flag bit that can be read, is cleared by writing a one, writing 0 has no effect.
Reset Value	
0	Resets to zero.
1	Resets to one.
—	Not defined after reset and not affected by reset.

All registers except the [Message Buffer Cycle Counter Filter Registers \(FR_MBCCFRn\)](#), [Message Buffer Frame ID Registers \(FR_MBFIDRn\)](#), and [Message Buffer Index Registers \(FR_MBIDXRn\)](#) are reset to their reset value on system reset. The registers mentioned above are located in physical memory blocks and, thus, they are not affected by reset. For some register fields, additional reset conditions exist. These additional reset conditions are described in the detailed description of the register. The additional reset conditions are explained in [Table 973](#).

Table 973. Additional register reset conditions

Condition	Description
Protocol RUN Command	The register field is reset when the application writes RUN command “0101” to the POCCMD field in the Protocol Operation Control Register (FR_POCR) .
Message Buffer Disable	The register field is reset when the application has disabled the message buffer. This happens when the application writes 1 to the message buffer disable trigger bit FR_MBCCSRn[EDT] while the message buffer is enabled (FR_MBCCSRn[EDS] = 1) and the CC grants a disable to the application by clearing the FR_MBCCSRn[EDS] bit.

51.5.2.2 Register write access

This section describes the write access restriction terms that apply to all registers.

51.5.2.2.1 Register write access restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in [Table 974](#). If, for a specific register field, none of the given write access conditions is fulfilled, any write attempt to this register field is ignored without any notification. The values of the fields are not changed. The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled. The condition term [A and B] indicates that the register or field can be written to if both conditions are fulfilled.

Table 974. Register Write Access Restrictions

Condition	Indication	Description
Anytime	—	No write access restriction.
Disabled mode	FR_MCR[MEN] = 0	Write access only when CC is in Disabled mode.
Normal mode	FR_MCR[MEN] = 1	Write access only when CC is in Normal mode.
POC:config	FR_PSR0[PROTSTATE] = <i>POC:config</i>	Write access only when protocol is in the <i>POC:config</i> state.
MB_DIS	FR_MBCCSR[EDS] = 0	Write access only when related Message Buffer is disabled.
MB_LCK	FR_MBCCSRn[LCKS] = 1	Write access only when related Message Buffer is locked.
IDL	FR_EEIRICR[BSY] = 0	Write access only when ECC configuration is idle.
Functional Test Mode	FR_MCR[TMODE] = 1	Write access only when CC is in Functional Test Mode.

51.5.2.2.2 Register Write Access Requirements

All registers can be accessed with 8-bit, 16-bit, and 32-bit wide operations.

For some of the registers, at least a 16-bit wide write access is required to ensure correct operation. This write access requirement is stated in the detailed register description for each register affected. If an 8-bit wide write access is performed to any of these registers, this access is ignored without notification.

51.5.2.2.3 Internal Register Access

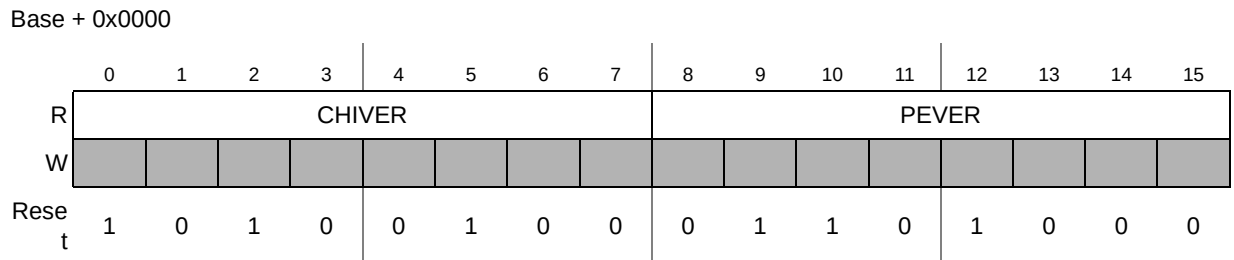
The following memory-mapped registers are used to access multiple internal registers.

- [Strobe Signal Control Register \(FR_STBSCR\)](#)
- [Slot Status Selection Register \(FR_SSSR\)](#)
- [Slot Status Counter Condition Register \(FR_SSCCR\)](#)
- Receive Shadow Buffer Configuration Data

Each of these memory-mapped registers provides a SEL field and a WMD bit. The SEL field is used to select the internal register. The WMD bit controls the write mode. If WMD is set to 0 during the write access, all fields of the internal register are updated. If WMD is set to 1, only the SEL field is changed. All other fields of the internal register remain unchanged. This allows for reading back the values of the selected internal register in a subsequent read access.

51.5.2.3 Module Version Register (FR_MVR)

Figure 940. Module Version Register (FR_MVR)



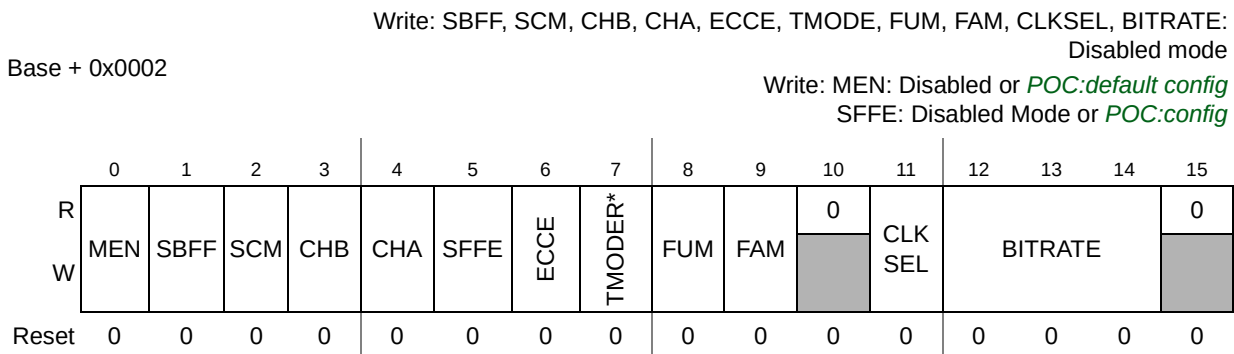
This register provides the CC version number. The module version number is derived from the CHI version number and the PE version number.

Table 975. FR_MVR field descriptions

Field	Description
CHIVER	CHI Version Number —This field provides the version number of the controller host interface.
PEVER	PE Version Number —This field provides the version number of the protocol engine.

51.5.2.4 Module Configuration Register (FR_MCR)

Figure 941. Module Configuration Register (FR_MCR)



This register defines the global configuration of the CC.

Table 976. FR_MCR field descriptions

Field	Description
MEN	Module Enabl—This bit indicates whether or not the CC is in Disabled mode. The application requests that the CC leave Disabled mode by writing 1 to this bit. Before leaving Disabled mode, the application must configure the SCM, SBFF, CHB, CHA, TMODE, and BITRATE values. For details, see Section 51.1.6, “Modes of operation”. 0 Write: only during <i>POC:default config</i>, CC disable Read: CC disabled 1 Write: enable CC Read: CC enabled <i>Note: If the CC is enabled, it can only be disabled during mode: POC:default config</i>
SBFF	System Bus Failure Freeze—This bit controls the behavior of the CC in case of a system bus failure. 0 Continue normal operation 1 Transition to Freeze mode
SCM	Single Channel Device Mode—This control bit defines the channel device mode of the CC as described in Section 51.6.10, “Channel device modes”. 0 CC works in Dual-Channel Device mode 1 CC works in Single-Channel Device mode
CHB CHA	Channel Enable—protocol related parameter: <i>pChannels</i> The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given Table 977.
SFFE	Synchronization Frame Filter Enable—This bit controls the filtering for received synchronization frames. For details see Section 51.6.15, “Sync frame filtering”. 0 Synchronization frame filtering disabled 1 Synchronization frame filtering enabled
ECCE	ECC Functionality Enable—This bit controls the ECC memory error detection functionality. For details see Section 51.6.24, “Memory content error detection”. 0 ECC functionality (injection, detection, reporting, response) disabled 1 ECC functionality enabled
TMODE	Test Mode—This control bit is used to enable and disable the functional test mode. 0 Functional Test Mode disabled 1 Functional Test Mode enabled
FUM	FIFO Update Mode—This bit controls the FIFO update behavior when the interrupt flags FR_GIFER[FAFAIF] and FR_GIFER[FAFBIF] are written by the application (see Section 51.6.9.8, “FIFO update”). 0 FIFOA/FIFOB is updated on writing 1 to FR_GIFER[FAFAIF] /FR_GIFER[FAFBIF] 1 FIFOA/FIFOB is <i>not</i> updated on writing 1 to FR_GIFER[FAFAIF]/FR_GIFER[FAFBIF]
FAM	FIFO Address Mode—This bit controls the location of the system memory base address for the FIFOs (see Section 51.6.9.2, “FIFO configuration”). 0 FIFO Base Address located in System Memory Base Address Register (FR_SYMBADR) 1 FIFO Base Address located in Receive FIFO System Memory Base Address Register (FR_RFSYMBADR)

Table 976. FR_MCR field descriptions

Field	Description
CLKSEL	Protocol Engine Clock Source Select —This bit is used to select the clock source for the protocol engine. 0 PE clock source is generated by on-chip crystal oscillator. 1 PE clock source is generated by on-chip PLL.
BITRATE	FlexRay Bus Bit Rate —This field defines the FlexRay bus bit rate. 000 10.0 Mbit/s 001 5.0 Mbit/s 010 2.5 Mbit/s 011 8.0 Mbit/s 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 977. FlexRay channel selection

SCM	CHB	CHA	Description
Dual Channel Device modes			
0	0	0	ports FR_A_RX, FR_A_TX, and FR_A_TX_EN not driven by CC ports FR_B_RX, FR_B_TX, and FR_A_TX_EN not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and FR_A_TX_EN driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and FR_A_TX_EN not driven by CC
	1	0	ports FR_A_RX, FR_A_TX, and FR_A_TX_EN not driven by CC ports FR_B_RX, FR_B_TX, and FR_A_TX_EN driven by CC—connected to FlexRay channel B
	1	1	ports FR_A_RX, FR_A_TX, and FR_A_TX_EN driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and FR_A_TX_EN driven by CC—connected to FlexRay channel B
Single Channel Device mode			
1	0	0	ports FR_A_RX, FR_A_TX, and FR_A_TX_EN not driven by CC ports FR_B_RX, FR_B_TX, and FR_A_TX_EN not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and FR_A_TX_EN driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and FR_A_TX_EN not driven by CC
	1	0	ports FR_A_RX, FR_A_TX, and FR_A_TX_EN driven by CC—connected to FlexRay channel B ports FR_B_RX, FR_B_TX, and FR_A_TX_EN not driven by CC
	1	1	Reserved

51.5.2.5 System Memory Base Address Register (FR_SYMBADR)

Figure 942. System Memory Base Address High Register (FR_SYMBADHR)

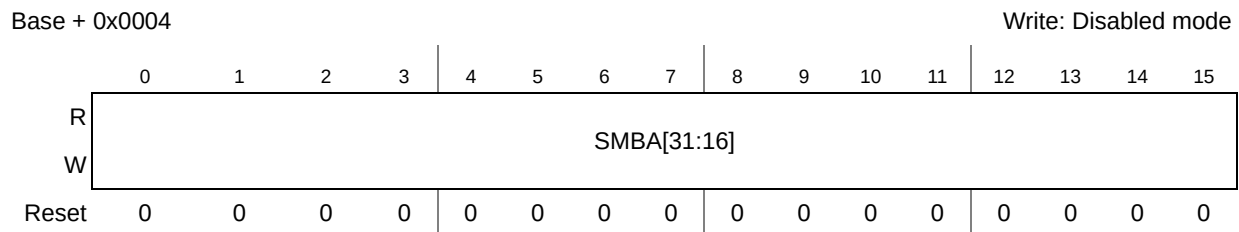
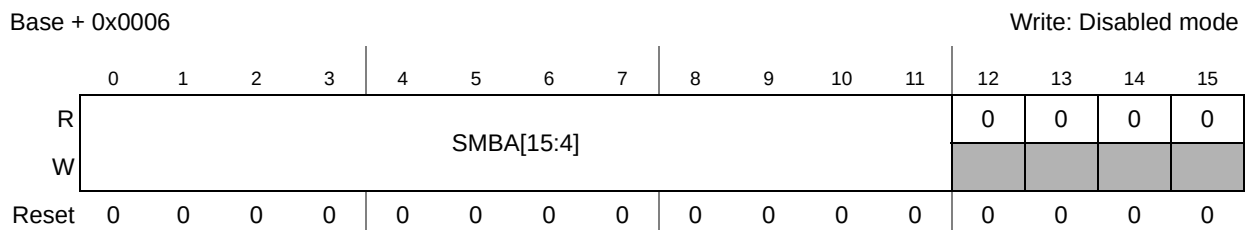


Figure 943. System Memory Base Address Low Register (FR_SYMBADLR)



Note: The system memory base address must be set before the CC is enabled.

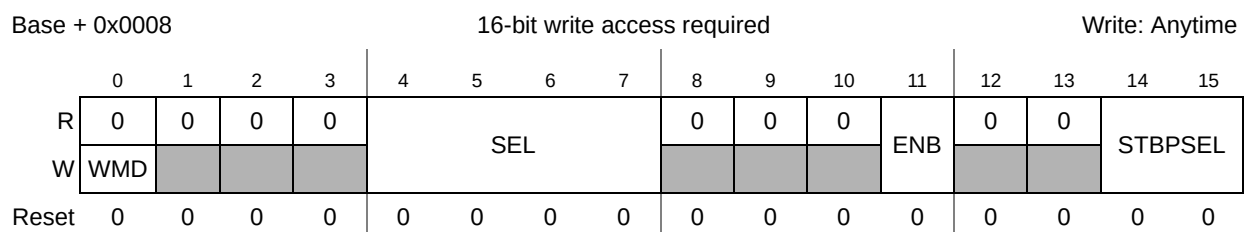
The system memory base address registers define the base address of the FlexRay memory area within the system memory. The base address is used by the bus master interface (BMIF) to calculate the physical memory address for system memory accesses.

Table 978. FR_SYMBADR field descriptions

Field	Description
SMBA	System Memory Base Address —This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if FR_MCR[FAM] is set to 1. It defines as a byte address.

51.5.2.6 Strobe Signal Control Register (FR_STBSCR)

Figure 944. Strobe Signal Control Register (FR_STBSCR)



This register is used to assign the individual protocol timing-related strobe signals given in [Table 980](#) to the external strobe ports. Each strobe signal can be assigned to at most one strobe port. Each write access to registers overwrites the previously-written ENB and STBPSEL values for the signal indicated by SEL. If more than one strobe signal is assigned to one strobe port, the current values of the strobe signals are combined with a binary OR

and presented at the strobe port. If no strobe signal is assigned to a strobe port, the strobe port carries logic 0. For more detailed and timing information refer to [Section 51.6.16](#), “Strobe signal support.”

Note: In Single-Channel Device mode, channel B-related strobe signals are undefined and should not be assigned to the strobe ports.

Table 979. FR_STBSCR field descriptions

Field	Description
WMD	Write Mode —This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
SEL	Strobe Signal Select —This control field selects one of the strobe signals given in Table 980 to be enabled or disabled and assigned to one of the four strobe ports given in Table 980 .
ENB	Strobe Signal Enable —The control bit is used to enable and disable the strobe signal selected by STBSSEL. 0 Strobe signal is disabled and not assigned to any strobe port. 1 Strobe signal is enabled and assigned to the strobe port selected by STBPSEL.
STBPSEL	Strobe Port Select —This field selects the strobe port that the strobe signal selected by the SEL is assigned to. All strobe signals that are enabled and assigned to the same strobe port are combined with a binary OR operation. 00 Assign selected signal to FR_DBG[0]. 01 Assign selected signal to FR_DBG[1]. 10 Assign selected signal to FR_DBG[2]. 11 Assign selected signal to FR_DBG[3].

Table 980. Strobe signal mapping

SEL		Description	Channel	Type	Offset ⁽¹⁾	Reference
dec	hex					
0	0x0	Arm	—	value	+1	MT start
1	0x1	mt	—	value	+1	MT start
2	0x2	Cycle start	—	pulse	0	MT start
3	0x3	Minislot start	—	pulse	0	MT start
4	0x4	Slot start	A	pulse	0	MT start
5	0x5		B			
6	0x6	Receive data after glitch filtering	A	value	+4	FR_A_RX
7	0x7		B			FR_B_RX
8	0x8	Channel idle indicator	A	level	+5	FR_A_RX
9	0x9		B			FR_B_RX
10	0xA	Syntax error detected	A	pulse	+4	FR_A_RX
11	0xB		B			FR_B_RX
12	0xC	Content error detected	A	level	+4	FR_A_RX
13	0xD		B			FR_B_RX

Table 980. Strobe signal mapping

SEL		Description	Channel	Type	Offset ⁽¹⁾	Reference
dec	hex					
14	0xE	Receive FIFO almost-full interrupt signals	A	value	n.a.	RX FIFO A Almost Full Interrupt
15	0xF		B			RX FIFO B Almost Full Interrupt

1. Given in PE clock cycles

51.5.2.7 Message Buffer Data Size Register (FR_MBDSR)

Figure 945. Message Buffer Data Size Register (FR_MBDSR)

Base + 0x000C

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines the size of the message buffer data section for the two message buffer segments in a number of two-byte entities.

The CC provides two independent segments for the individual message buffers. All individual message buffers within one segment have to have the same size for the message buffer data section. This size can be different for the two message buffer segments.

Table 981. FR_MBDSR field descriptions

Field	Description
MBSEG2DS	Message Buffer Segment 2 Data Size —The field defines the size of the message buffer data section in two-byte entities for message buffers within the <i>second</i> message buffer segment.
MBSEG1DS	Message Buffer Segment 1 Data Size —The field defines the size of the message buffer data section in two-byte entities for message buffers within the <i>first</i> message buffer segment.

51.5.2.8 Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

Figure 946. Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

Base + 0x000E

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

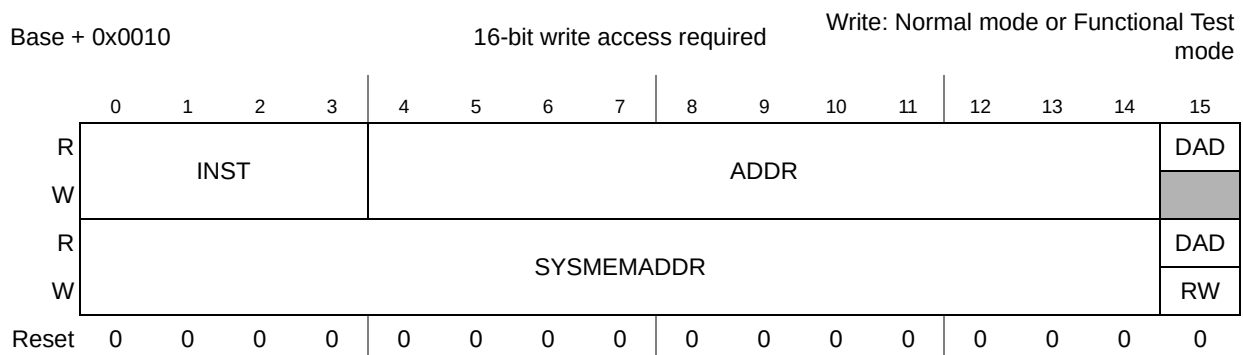
This register defines the last individual message buffer that belongs to the first message buffer segment and the number of the last used individual message buffer.

Table 982. FR_MBSSUTR Field Descriptions

Field	Description
LAST_MB_SEG1	Last Message Buffer In Segment 1—This field defines the message buffer number of the last individual message buffer that is assigned to the <i>first</i> message buffer segment. The individual message buffers in the <i>first</i> segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, and FR_MBIDXRn with $n \leq \text{LAST_MB_SEG1}$. The first message buffer segment contains $\text{LAST_MB_SEG1} + 1$ individual message buffers. <i>Note: The first message buffer segment contains at least one individual message buffer.</i> The individual message buffers in the <i>second</i> message buffer segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with $\text{LAST_MB_SEG1} < n < 128$. <i>Note: If $\text{LAST_MB_SEG1} = 127$, all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.</i>
LAST_MB_UTIL	Last Message Buffer Utilized—This field defines the message buffer number of last utilized individual message buffer. The message buffer search engine examines all individual message buffer with a message buffer number $n \leq \text{LAST_MB_UTIL}$. <i>Note: If $\text{LAST_MB_UTIL} = \text{LAST_MB_SEG1}$, all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.</i>

51.5.2.9 PE DRAM Access Register (FR_PEDRAR)

Figure 947. PE DRAM Access Register (FR_PEDRAR)



In Functional Test mode this register is used to setup read and write operations to the system memory.

The FR_PEDRAR is used to trigger write and read operations on the PE data memory (PE DRAM). These operations are used for memory error injection and memory error observation.

Each write access to this register initiates a read or write operation on the PE DRAM. The access done status bit, DAD, is cleared after the write access and is set if the PE DRAM access has been finished.

In case of an PE DRAM write access, the data provided in FR_PEDRAR is written into the PE DRAM, read back from the PE DRAM, and are stored into the FR_PEDRAR.

In case of an PE DRAM read access, the requested data is read from PE DRAM and stored into the FR_PEDRAR.

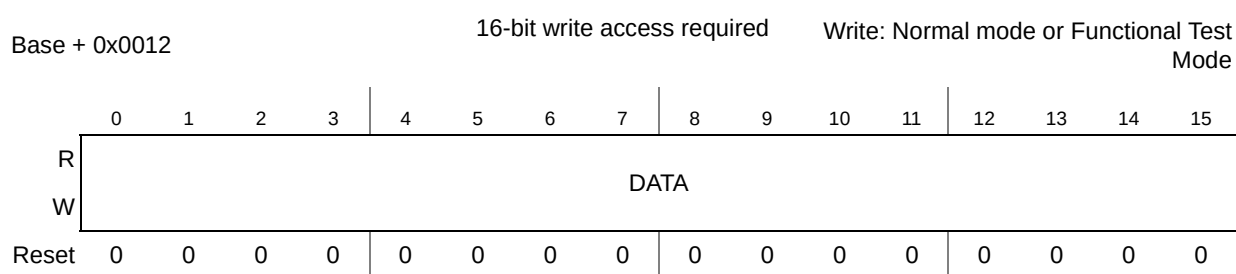
For a detailed description, refer to [Section 51.6.24, “Memory content error detection”](#)

Table 983. FR_PEDRAR field descriptions

Field	Description
INST	PE DRAM Access Instruction —This field defines the operation to be executed on the PE DRAM. 0011 PE DRAM write: Write FR_PEDRDR[DATA] to PE DRAM address ADDR (16 bit) 0101 PE DRAM read: Read Data from PE DRAM address ADDR (16 bit) into FR_PEDRDR[DATA] other reserved
ADDR	PE DRAM Access Address —This field defines the address in the PE DRAM to be written to or read from.
DAD	PE DRAM Access Done —This status bit is cleared when the application has written to this register and is set when the PE DRAM access has finished. 0 PE DRAM access running 1 PE DRAM access done
SYSTEMADDR	Functional Test Mode Only System Memory Address — This field defines the system memory address to be written to or read from.
RW	Functional Test Mode Only Debug Read Write Selector — This control bit defines the access direction for the system memory access. 0 system memory write access 1 system memory read access

51.5.2.10 PE DRAM Data Register (FR_PEDRDR)

Figure 948. PE DRAM Data Register (FR_PEDRDR)



This register provides the data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

51.5.2.11 Protocol Operation Control Register (FR_POCR)

Figure 949. Protocol Operation Control Register (FR_POCR)

Base + 0x0014												Write: Normal mode				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	EOC_AP		ERC_AP		BSY	0	0	0	POCCMD			
W	WME								WMC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The application uses this register to issue:

- Protocol control commands
- External clock correction commands

Protocol control commands are issued by writing to the POCCMD field. For more information on protocol control commands, see [Section 51.7.6, “Protocol Control Command Execution”](#).

External clock correction commands are issued by writing to the EOC_AP and ERC_AP fields. For more information on external clock correction, refer to [Section 51.6.11, “External clock synchronization”](#).

Table 984. FR_POCR Field Descriptions

Field	Description
WME	Write Mode External Correction —This bit controls the write mode of the EOC_AP and ERC_AP fields. 0 Write to EOC_AP and ERC_AP fields on register write 1 No write to EOC_AP and ERC_AP fields on register write
EOC_AP	External Offset Correction Application —This field is used to trigger the application of the external offset correction value defined in the Protocol Configuration Register 29 (FR_PCR29) . 00 Do not apply external offset correction value. 01 Reserved. 10 Subtract external offset correction value 11 Add external offset correction value
ERC_AP	External Rate Correction Application —This field is used to trigger application of the external rate correction value defined in the Protocol Configuration Register 21 (FR_PCR21) . 00 Do not apply external rate correction value 01 Reserved 10 Subtract external rate correction value 11 Add external rate correction value

Table 984. FR_POCR Field Descriptions

Field	Description
BSY	Protocol Control Command Write Busy —This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the <i>CHI Error Flag Register (FR_CHIERFR)</i> , and will not change the value of the POCCMD field. 0 Command write idle, command accepted and ready to receive new protocol command 1 Command write busy, command not yet accepted, not ready to receive new protocol command
WMC	Write Mode Command —This bit controls the write mode of the POCCMD field. 0 Write to POCCMD field on register write 1 Do not write to POCCMD field on register write
POCCMD	Protocol Control Command —The application writes to this field to issue a protocol control command to the PE. The CC sends the protocol command to the PE immediately. While the transfer is running, the BSY bit is set. 0000 ALLOW_COLDSTAR—Immediately activate capability of node to cold start cluster. 0001 ALL_SLOTS—Delayed ⁽¹⁾ transition to the All Slots Transmission mode. 0010 CONFIG—Immediately transition to the <i>POC:config</i> state. 0011 FREEZE—Immediately transition to the <i>POC:halt</i> state. 0100 READY, CONFIG_COMPLETE—Immediately transition to the <i>POC:ready</i> state. 0101 RUN—Immediately transition to the <i>POC:startup start</i> state. 0110 DEFAULT_CONFIG—Immediately transition to the <i>POC:default config</i> state. 0111 HALT—Delayed transition to the <i>POC:halt</i> state 1000 WAKEUP—Immediately initiate the wakeup procedure. 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

1. Delayed means on completion of current communication cycle.

51.5.2.12 Global Interrupt Flag and Enable Register (FR_GIFER)

Figure 950. Global Interrupt Flag and Enable Register (FR_GIFER)

Base + 0x0016				In Functional Test Mode writing 1 toggles the interrupt flags.								Write: Normal mode				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIF	PRIF	CHIF	WUIF	FAFBIF	FAFAIF	RBIF	TBIF	MIE	PRIE	CHIE	WUIE	FAFBIE	FAFAIE	RBIE	TBIE
W				w1c	w1c	w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register controls some of the interrupt request lines and provides the corresponding interrupt flags. The interrupt flags MIF, PRIF, CHIF, RBIF, and TBIF are the outcome of a binary OR of the related individual interrupt flags and interrupt enables. The generation

scheme for these flags is depicted in [Figure 1098](#). For more details on interrupt generation, see [Section 51.6.20, “Interrupt support](#). These flags are cleared automatically when all of the corresponding interrupt flags or interrupt enables in the related interrupt flag and enable registers are cleared by the application.

Table 985. FR_GIFER field descriptions

Field	Description
MIF	Module Interrupt Flag —This interrupt flag is set if at least one of the other interrupt flags in this register and the related interrupt enable bit are set. 0 No interrupt flag and related interrupt enable bit are set 1 At least one of the other interrupt flags in this register and the related interrupt bit are set
PRIF	Protocol Interrupt Flag —This interrupt flag is set if at least one of the individual flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1) and the related interrupt enable bit are set. 0 No individual protocol interrupt flag and related interrupt enable bit are set 1 At least one of the individual protocol interrupt flags and the related interrupt enable bit are set
CHIF	CHI Interrupt Flag —This interrupt flag is set if at least one of the error flags in the CHI Error Flag Register (FR_CHIERFR) and the CHI error interrupt enable bit FR_GIFER[CHIE] are set. 0 All CHI error flags are equal to 0 or the CHI error interrupt is disabled 1 At least one CHI error flag and the CHI error interrupt enable are is set
WUPIF	Wakeup Interrupt Flag —This interrupt flag is set when the CC has received a wakeup symbol on the FlexRay bus. The application can determine on which channel the wakeup symbol was received by reading the related wakeup flags WUB and WUA in the Protocol Status Register 3 (FR_PSR3) . 0 No wakeup symbol received on FlexRay bus 1 Wakeup symbol received on FlexRay bus
FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag —This interrupt flag is set when one of the following events occurs: a) The current number of FIFO B entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR) , and the CC writes a received message into the FIFO B, or b) The current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 No such event 1 FIFO B almost full event has occurred
FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag —This interrupt flag is set when one of the following events occurs: a) The current number of FIFO A entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR) , and the CC writes a received message into the FIFO A, or b) The current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 No such event 1 FIFO A almost full event has occurred

Table 985. FR_GIFER field descriptions (continued)

Field	Description
RBIF	Receive Message Buffer Interrupt Flag—This interrupt flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) both the MBIF interrupt flag and the MBIE interrupt enable bit in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are asserted. The application can not clear this interrupt flag directly; instead, it is cleared by the CC when all of the MBIF interrupt flags of the individual receive message buffers are cleared by the application, or when the application has cleared the related MBIE interrupt enable bit. 0 None of the individual receive message buffers has the MBIF and MBIE flag set 1 At least one individual receive message buffer has the MBIF and MBIE flag set
TBIF	Transmit Message Buffer Interrupt Flag—This flag is set if for at least one of the individual message buffers (FR_MBCCSRn[MTD] = 1) both the MBIF interrupt flag and the MBIE interrupt enable bit in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are equal to 1. The application cannot clear this interrupt flag directly; instead, this interrupt flag is cleared by the CC when all of the individual interrupt flags MBIF of the individual transmit message buffers are cleared by the application, or when the application has cleared the related MBIE interrupt enable bit. 0 None of the individual transmit message buffers has the MBIF and MBIE flag set 1 At least one individual transmit message buffer has the MBIF and MBIE flag set
MIE	Module Interrupt Enable—This bit controls whether the Module Interrupt line is asserted when the MIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
PRIE	Protocol Interrupt Enable—This bit controls whether the Protocol Interrupt line is asserted when the PRIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
CHIE	CHI Interrupt Enable—This bit controls whether the CHI Interrupt line is asserted when the CHIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
WUPIE	Wakeup Interrupt Enable—This bit controls whether the Wakeup Interrupt line is asserted when the WUPIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
FAFBIE	Receive FIFO Channel B Almost Full Interrupt Enable—This bit controls whether the RX FIFO B Almost Full Interrupt line is asserted when the FAFBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
FAFAIE	Receive FIFO Channel A Almost Full Interrupt Enable—This bit controls whether the RX FIFO A Almost Full Interrupt line is asserted when the FAFAIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

Table 985. FR_GIFER field descriptions (continued)

Field	Description
RBIE	Receive Message Buffer Interrupt Enable —This bit controls whether the Receive Message Buffer Interrupt line is asserted when the RBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
TBIE	Transmit Message Buffer Interrupt Enable —This bit controls whether the Transmit Message Buffer Interrupt line is asserted when the TBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

51.5.2.13 Protocol Interrupt Flag Register 0 (FR_PIFR0)

Figure 951. Protocol Interrupt Flag Register 0 (FR_PIFR0)

Base + 0x0018				In Functional Test Mode writing 1 toggles the interrupt flags.								Write: Normal mode				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FATL _IF	INTL _IF	ILCF _IF	CSA _IF	MRC _IF	MOC _IF	CCL _IF	MXS _IF	MTX _IF	LTXB _IF	LTX A_IF	TBVB _IF	TBVA _IF	TI2 _IF	TI1 _IF	CYS _IF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register holds one set of the protocol-related individual interrupt flags.

Table 986. FR_PIFR0 field descriptions

Field	Description
FATL_IF	Fatal Protocol Error Interrupt Flag —This flag is set when the protocol engine has detected a fatal protocol error. In this case, the protocol engine goes into the <i>POC:halt</i> state immediately. The fatal protocol errors are: 1) <i>pLatestTx</i> violation, as described in the MAC process of the FlexRay protocol 2) transmission across slot boundary violation, as described in the FSP process of the FlexRay protocol 0 No such event 1 Fatal protocol error detected
INTL_IF	Internal Protocol Error Interrupt Flag —This flag is set when the protocol engine has detected an internal protocol error. In this case, the protocol engine goes into the <i>POC:halt</i> state immediately. An internal protocol error occurs when the protocol engine has not finished a calculation and a new calculation is requested. This can be caused by a hardware error. 0 No such event 1 Internal protocol error detected

Table 986. FR_PIFR0 field descriptions (continued)

Field	Description
ILCF_IF	Illegal Protocol Configuration Interrupt Flag—This flag is set when the protocol engine has detected an illegal protocol configuration parameter setting. In this case, the protocol engine goes into the <i>POC:halt</i> state immediately. The protocol engine checks the <i>listen_timeout</i> value programmed into the Protocol Configuration Register 14 (FR_PCR14) and Protocol Configuration Register 15 (FR_PCR15) when the CONFIG_COMPLETE command was sent by the application via the Protocol Operation Control Register (FR_POCR). If the value of <i>listen_timeout</i> is equal to 0, the protocol configuration setting is considered as illegal. 0 No such event 1 Illegal protocol configuration detected
CSA_IF	Cold Start Abort Interrupt Flag—This flag is set when the configured number of allowed cold start attempts is reached and none of these attempts was successful. The number of allowed cold start attempts is configured by the <i>coldstart_attempts</i> field in the Protocol Configuration Register 3 (FR_PCR3). 0 No such event 1 Cold start aborted and no more coldstart attempts allowed
MRC_IF	Missing Rate Correction Interrupt Flag—This flag is set when an insufficient number of measurements is available for rate correction at the end of the communication cycle. 0 No such event 1 Insufficient number of measurements for rate correction detected
MOC_IF	Missing Offset Correction Interrupt Flag—This flag is set when an insufficient number of measurements is available for offset correction. This is related to the MISSING_TERM event in the CSP process for offset correction in the FlexRay protocol. 0 No such event 1 Insufficient number of measurements for offset correction detected
CCL_IF	Clock Correction Limit Reached Interrupt Flag—This flag is set when the internal calculated offset or rate calculation values have reached or exceeded their configured thresholds as given by the <i>offset_coorection_out</i> field in the Protocol Configuration Register 9 (FR_PCR9) and the <i>rate_correction_out</i> field in the Protocol Configuration Register 14 (FR_PCR14). 0 No such event 1 Offset or rate correction limit reached
MXS_IF	Max Sync Frames Detected Interrupt Flag—This flag is set when the number of synchronization frames detected in the current communication cycle exceeds the value of the <i>node_sync_max</i> field in the Protocol Configuration Register 30 (FR_PCR30). 0 No such event. 1 More than <i>node_sync_max</i> sync frames detected <i>Note: Only synchronization frames that have passed the synchronization frame acceptance and rejection filters are taken into account</i>
MTX_IF	Media Access Test Symbol Received Interrupt Flag—This flag is set when the MTS symbol was received on channel A or channel B. 0 No such event 1 MTS symbol received
LTXB_IF	pLatestTx Violation on Channel B Interrupt Flag—This flag is set when the frame transmission on channel B in the dynamic segment exceeds the dynamic segment boundary. This is related to the <i>pLatestTx</i> violation, as described in the MAC process of the FlexRay protocol. 0 No such event 1 <i>pLatestTx</i> violation occurred on channel B

Table 986. FR_PIFR0 field descriptions (continued)

Field	Description
LTXA_IF	<i>pLatestTx</i> Violation on Channel A Interrupt Flag —This flag is set when the frame transmission on channel A in the dynamic segment exceeds the dynamic segment boundary. This is related to the <i>pLatestTx</i> violation as described in the MAC process of the FlexRay protocol. 0 No such event 1 <i>pLatestTx</i> violation occurred on channel A
TBVB_IF	Transmission across boundary on channel B Interrupt Flag —This flag is set when the frame transmission on channel B crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event 1 Transmission across boundary violation occurred on channel B
TBVA_IF	Transmission across boundary on channel A Interrupt Flag —This flag is set when the frame transmission on channel A crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event 1 Transmission across boundary violation occurred on channel A
TI2_IF	Timer 2 Expired Interrupt Flag —This flag is set whenever timer 2 expires. 0 No such event 1 Timer 2 has reached its time limit
TI1_IF	Timer 1 Expired Interrupt Flag —This flag is set whenever timer 1 expires. 0 No such event 1 Timer 1 has reached its time limit
CYS_IF	Cycle Start Interrupt Flag —This flag is set when a communication cycle starts. 0 No such event 1 Communication cycle started.

51.5.2.14 Protocol Interrupt Flag Register 1 (FR_PIFR1)

Figure 952. Protocol Interrupt Flag Register 1 (FR_PIFR1)

Base + 0x001A				In Functional Test Mode writing 1 toggles the interrupt flags.								Write: Normal mode				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMC_IF	IPC_IF	PEC_F_IF	PSC_IF	SSI3_IF	SSI2_IF	SSI1_IF	SSI0_IF	0	0	EVT_IF	ODT_IF	0	0	0	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The register holds one set of the protocol-related individual interrupt flags.

Table 987. FR_PIFR1 field descriptions

Field	Description
EMC_IF	Error Mode Changed Interrupt Flag —This flag is set when the value of the ERRMODE bit field in the Protocol Status Register 0 (FR_PSR0) is changed by the CC. 0 No such event 1 ERRMODE field changed
IPC_IF	Illegal Protocol Control Command Interrupt Flag —This flag is set when the PE tries to execute a protocol control command that was issued via the POCCMD field of the Protocol Operation Control Register (FR_POCR) , and detects that this protocol control command is not allowed in the current protocol state. In this case, the command is not executed. For more details, see Section 51.7.6, “Protocol Control Command Execution” . 0 No such event 1 Illegal protocol control command detected
PECF_IF	Protocol Engine Communication Failure Interrupt Flag —This flag is set if the CC has detected a communication failure between the PE and the CHI. 0 No such event 1 Protocol Engine Communication Failure detected
PSC_IF	Protocol State Changed Interrupt Flag —This flag is set when the protocol state in the PROTSTATE field in the Protocol Status Register 0 (FR_PSR0) has changed. 0 No such event 1 Protocol state changed
SSI3_IF SSI2_IF SSI1_IF SSI0_IF	Slot Status Counter Incremented Interrupt Flag —Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCR0–FR_SSCR3) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
EVT_IF	Even Cycle Table Written Interrupt Flag —This flag is set if the CC has written the sync frame measurement/ID tables into the FlexRay memory area for the even cycle. 0 No such event 1 Sync frame measurement table written
ODT_IF	Odd Cycle Table Written Interrupt Flag —This flag is set if the CC has written the sync frame measurement/ID tables into the FlexRay memory area for the odd cycle. 0 No such event 1 Sync frame measurement table written

51.5.2.15 Protocol Interrupt Enable Register 0 (FR_PIER0)

Figure 953. Protocol Interrupt Enable Register 0 (FR_PIER0)

Base + 0x001C

Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FATL	INTL	ILCF	CSA	MRC	MOC	CCL	MXS	MTX	LTXB	LTXA	TBVB	TBVA	TI2	TI1	CYS
W	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE	_IE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines whether or not the individual interrupt flags defined in the [Protocol Interrupt Flag Register 0 \(FR_PIFR0\)](#) can generate a protocol interrupt request.

Table 988. FR_PIER0 field descriptions

Field	Description
FATL_IE	Fatal Protocol Error Interrupt Enable —This bit controls FATL_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
INTL_IE	Internal Protocol Error Interrupt Enable —This bit controls INTL_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
ILCF_IE	Illegal Protocol Configuration Interrupt Enable —This bit controls ILCF_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
CSA_IE	Cold Start Abort Interrupt Enable —This bit controls CSA_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
MRC_IE	Missing Rate Correction Interrupt Enable —This bit controls MRC_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
MOC_IE	Missing Offset Correction Interrupt Enable —This bit controls MOC_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
CCL_IE	Clock Correction Limit Reached Interrupt Enable —This bit controls CCL_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
MXS_IE	Max Sync Frames Detected Interrupt Enable —This bit controls MXS_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
MTX_IE	Media Access Test Symbol Received Interrupt Enable —This bit controls MTX_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
LTXB_IE	<i>pLatestTx</i> Violation on Channel B Interrupt Enable —This bit controls LTXB_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled
LTXA_IE	<i>pLatestTx</i> Violation on Channel A Interrupt Enable —This bit controls LTXA_IF interrupt request generation. 0interrupt request generation disabled 1interrupt request generation enabled

Table 988. FR_PIER0 field descriptions

Field	Description
TBVB_IE	Transmission across boundary on channel B Interrupt Enable —This bit controls TBVB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
TBVA_IE	Transmission across boundary on channel A Interrupt Enable —This bit controls TBVA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
TI2_IE	Timer 2 Expired Interrupt Enable —This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
TI1_IE	Timer 1 Expired Interrupt Enable —This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
CYS_IE	Cycle Start Interrupt Enable —This bit controls CYC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

51.5.2.16 Protocol Interrupt Enable Register 1 (FR_PIER1)

Figure 954. Protocol Interrupt Enable Register 1 (FR_PIER1)

Base + 0x001E								Write: Anytime								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMC _IE	IPC _IE	PEC F_IE	PSC _IE	SSI3 _IE	SSI2 _IE	SSI1 _IE	SSI0 _IE	0	0	EVT _IE	ODT _IE	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines whether or not the individual interrupt flags defined in [Protocol Interrupt Flag Register 1 \(FR_PIFR1\)](#) can generate a protocol interrupt request.

Table 989. FR_PIER1 field descriptions

Field	Description
EMC_IE	Error Mode Changed Interrupt Enable —This bit controls EMC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
IPC_IE	Illegal Protocol Control Command Interrupt Enable —This bit controls IPC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table 989. FR_PIER1 field descriptions

Field	Description
PECF_IE	Protocol Engine Communication Failure Interrupt Enable —This bit controls PECF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
PSC_IE	Protocol State Changed Interrupt Enable —This bit controls PSC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
SSI3_IE SSI2_IE SSI1_IE SSI0_IE	Slot Status Counter Incremented Interrupt Enable —This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
EVT_IE	Even Cycle Table Written Interrupt Enable —This bit controls EVT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
ODT_IE	Odd Cycle Table Written Interrupt Enable —This bit controls ODT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

51.5.2.17 CHI Error Flag Register (FR_CHIERFR)

Figure 955. CHI Error Flag Register (FR_CHIERFR)

Base + 0x0020				In Functional Test Mode writing 1 toggles the interrupt flags.								Write: Normal mode				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRLB _EF	FRLA _EF	PCMI _EF	FOV B_EF	FOV A_EF	MBS _EF	MBU _EF	LCK _EF	0	SBC F_EF	FID _EF	DPL _EF	SPL _EF	NML _EF	NMF _EF	ILSA _EF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register holds the CHI-related error flags. The interrupt generation for each of these error flags is controlled by the CHI interrupt enable bit, CHIE, in the [Global Interrupt Flag and Enable Register \(FR_GIFER\)](#).

Table 990. FR_CHIERFR field descriptions

Field	Description
FRLB_EF	Frame Lost Channel B Error Flag—This flag is set if a complete frame was received on channel B but it could not be stored in the selected individual message buffer because the message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such event 1 Frame lost on channel B detected
FRLA_EF	Frame Lost Channel A Error Flag—This flag is set if a complete frame was received on channel A but it could not be stored in the selected individual message buffer because the message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such error 1 Frame lost on channel A detected
PCMI_EF	Protocol Command Ignored Error Flag—This flag is set if the application has issued a POC command by writing to the POCCMD field in the Protocol Operation Control Register (FR_POCCR) while the BSY flag is equal to 1. In this case, the command is ignored by the CC and is lost. 0 No such error 1 POC command ignored
FOVB_EF	Receive FIFO Overrun Channel B Error Flag—This flag is set when an overrun of the FIFO for channel B has occurred. This error occurs if a semantically valid frame was received on channel B and matches all criteria to be appended to the FIFO for channel B but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
FOVA_EF	Receive FIFO Overrun Channel A Error Flag—This flag is set when an overrun of the FIFO for channel A has occurred. This error occurs if a semantically valid frame was received on channel A and matches the all criteria to be appended to the FIFO for channel A but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
MBS_EF	Message Buffer Search Error Flag—This flag is set if at least one of the following events occurs: a) The message buffer search engine is still running while the next search must be started due to the FlexRay protocol timing. b) A message buffer index greater than 131 is detected in the FR_MBIDXR[MBIDX] field of a found message buffer or in one of the FR_RSBIR[RSBIDX] fields. Refer to Section 51.6.7.4, "Message buffer search error" for details. 0 No such event 1 Search engine active while search start appears or illegal message buffer index detected
MBU_EF	Message Buffer Utilization Error Flag—This flag is asserted if the application writes to a message buffer control field that is beyond the number of utilized message buffers programmed in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). If the application writes to a FR_MBCCSRn register with n > LAST_MB_UTIL, the CC ignores the write attempt and asserts the message buffer utilization error flag MBU_EF in the CHI Error Flag Register (FR_CHIERFR). 0 No such event 1 Non-utilized message buffer enabled

Table 990. FR_CHIERFR field descriptions (continued)

Field	Description
LCK_EF	Lock Error Flag —This flag is set if the application tries to lock a message buffer that is already locked by the CC due to internal operations. In that case, the CC does not grant the lock to the application. The application must issue the lock request again. 0 No such error 1 Lock error detected
SBCF_EF	System Bus Communication Failure Error Flag —This flag is set if a system bus access was not finished within the required amount of time (see Section 51.6.19.1.2, “System bus access timeout”). 0 No such event 1 System bus access not finished in time
FID_EF	Frame ID Error Flag —This flag is set if the frame ID stored in the message buffer header area differs from the frame ID stored in the message buffer control register. 0 No such error occurred 1 Frame ID error occurred
DPL_EF	Dynamic Payload Length Error Flag —This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the dynamic segment is greater than the maximum payload length for the dynamic segment as it is configured in the corresponding max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24) . 0 No such error occurred 1 Dynamic payload length error occurred
SPL_EF	Static Payload Length Error Flag —This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the static segment is different from the payload length for the static segment as it is configured in the corresponding payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19) . 0 No such error occurred 1 Static payload length error occurred
NML_EF	Network Management Length Error Flag —This flag is set if the payload length written into the header structure of a receive message buffer assigned to the static segment is less than the configured length of the Network Management Vector as configured in the Network Management Vector Length Register (FR_NMVLR) . In this case, the received part of the Network Management Vector will be used to update the Network Management Vector. 0 No such error occurred 1 Network management length error occurred
NMF_EF	Network Management Frame Error Flag —This flag is set if a received message in the static segment with a Preamble Indicator flag (PP) asserted has its Null Frame indicator flag (NF) asserted as well. In this case, the Global Network Management Registers (see Section 51.5.2.39, “Network Management Vector Registers (FR_NMVR0–FR_NMVR5)”) are not updated. 0 No such error occurred 1 Network management frame error occurred
ILSA_EF	Illegal System Bus Address Error Flag —This flag is set if the external system bus subsystem has detected an access to an illegal system bus address from the CC (see Section 51.6.19.1.1, “System bus illegal address access”). 0 No such event 1 Illegal system bus address accessed

51.5.2.18 Message Buffer Interrupt Vector Register (FR_MBIVEC)

Figure 956. Message Buffer Interrupt Vector Register (FR_MBIVEC)

Base + 0x0022

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TBIVEC							0	RBIVEC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register indicates the lowest numbered receive message buffer and the lowest numbered transmit message buffer that have their MBIF interrupt status flag and MBIE interrupt enable bits asserted. This means that message buffers with lower message buffer numbers have higher priority.

Table 991. FR_MBIVEC field descriptions

Field	Description
TBIVEC	Transmit Buffer Interrupt Vector —This field provides the number of the lowest numbered enabled transmit message buffer that has its MBIF interrupt status flag and its MBIE interrupt enable bit set. If there is no transmit message buffer with the MBIF interrupt status flag and the MBIE interrupt enable bits asserted, the value in this field is set to 0.
RBIVEC	Receive Buffer Interrupt Vector —This field provides the message buffer number of the lowest numbered receive message buffer that has its MBIF interrupt flag and its MBIE interrupt enable bit asserted. If there is no receive message buffer with the MBIF interrupt status flag and the MBIE interrupt enable bits asserted, the value in this field is set to 0.

51.5.2.19 Channel A Status Error Counter Register (FR_CASERCR)

Figure 957. Channel A Status Error Counter Register (FR_CASERCR)

Base + 0x0024

Additional Reset: RUN Command

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STATUS_ERR_CNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

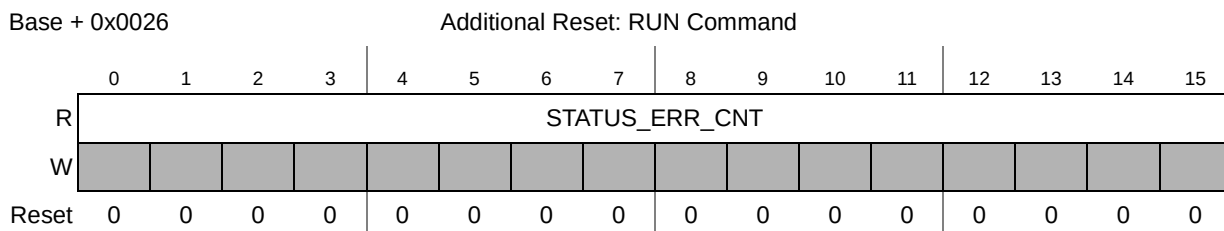
This register provides the channel status error counter for channel A. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits [vSS!SyntaxError](#), [vSS!ContentError](#), [vSS!BViolation](#), and [vSS!TxConflict](#). The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, see [Section 51.6.18, “Slot status monitoring”](#).

Table 992. FR_CASERCR Field Descriptions

Field	Description
STATUS_ERR_CNT	Channel Status Error Counter — This field provides the current value channel status error counter. The counter value is updated within the first macrotick of the following slot or segment.

51.5.2.20 Channel B Status Error Counter Register (FR_CBSECR)

Figure 958. Channel B Status Error Counter Register (FR_CBSECR)



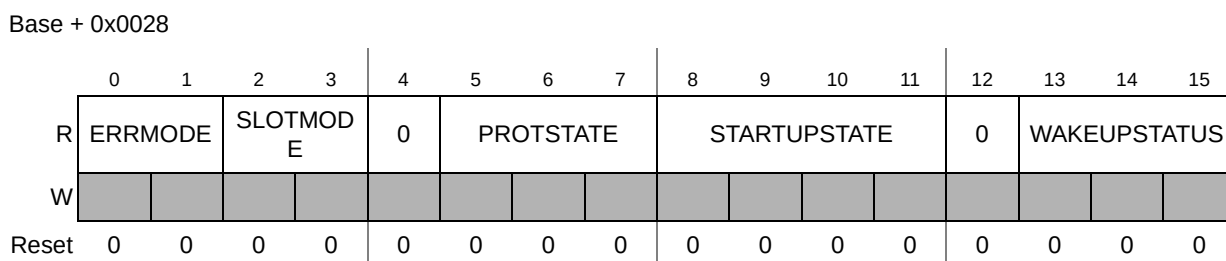
This register provides the channel status error counter for channel B. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the network idle time (NIT). The slot status vector contains the four protocol-related error indicator bits [vSS!SyntaxError](#), [vSS!ContentError](#), [vSS!BViolation](#), and [vSS!TxConflict](#). The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, see [Section 51.6.18, “Slot status monitoring”](#).

Table 993. FR_CBSECR field descriptions

Field	Description
STATUS_ERR_CNT	Channel Status Error Counter —This field provides the current channel status error count. The counter value is updated within the first macrotick of the following slot or segment.

51.5.2.21 Protocol Status Register 0 (FR_PSR0)

Figure 959. Protocol Status Register 0 (FR_PSR0)



This register provides information about the current protocol status.

Table 994. FR_PSR0 field descriptions

Field	Description
ERRMODE	Error Mode —protocol-related variable: <i>vPOC!ErrorMode</i> . This field indicates the error mode of the protocol. 00 ACTIVE 01 PASSIVE 10 COMM_HALT 11 Reserved
SLOTMODE	Slot Mode —protocol-related variable: <i>vPOC!SlotMode</i> . This field indicates the slot mode of the protocol. 00 SINGLE 01 ALL_PENDING 10 ALL 11 Reserved
PROTSTATE	Protocol State —protocol-related variable: <i>vPOC!State</i> . This field indicates the state of the protocol. 000 <i>POC:default config</i> 001 <i>POC:config</i> 010 <i>POC:wakeup</i> 011 <i>POC:ready</i> 100 <i>POC:normal passive</i> 101 <i>POC:normal active</i> 110 <i>POC:halt</i> 111 <i>POC:startup</i>
STARTUP STATE	Startup State —protocol-related variable: <i>vPOC!StartupState</i> . This field indicates the current sub-state of the startup procedure. 0000 Reserved 0001 Reserved 0010 <i>POC:coldstart collision resolution</i> 0011 <i>POC:coldstart listen</i> 0100 <i>POC:integration consistency check</i> 0101 <i>POC:integrationi listen</i> 0110 Reserved 0111 <i>POC:initialize schedule</i> 1000 Reserved 1001 Reserved 1010 <i>POC:coldstart consistency check</i> 1011 Reserved 1100 Reserved 1101 <i>POC:integration coldstart check</i> 1110 <i>POC:coldstart gap</i> 1111 <i>POC:coldstart join</i>
WAKEUP STATUS	Wakeup Status —protocol-related variable: <i>vPOC!WakeupStatus</i> . This field provides the outcome of the execution of the wakeup mechanism. 000 UNDEFINED 001 RECEIVED_HEADER 010 RECEIVED_WUP 011 COLLISION_HEADER 100 COLLISION_WUP 101 COLLISION_UNKNOWN 110 TRANSMITTED 111 Reserved

51.5.2.22 Protocol Status Register 1 (FR_PSR1)

Figure 960. Protocol Status Register 1 (FR_PSR1)

Base + 0x002A				Additional Reset: CSAA, CSP, CPN: RUN Command								Write: Normal mode				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSAA	CSP	0	REMCSAT					CPN	HHR	FRZ		APTAC			
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 995. FR_PSR1 field descriptions

Field	Description
CSAA	Cold Start Attempt Aborted Flag —protocol-related event: ‘set coldstart abort indicator in CHI.’ This flag is set when the CC has aborted a cold start attempt. 0 No such event 1 Cold start attempt aborted
CSP	Leading Cold Start Path —This status bit is set when the CC has reached the <i>POC:normal active</i> state via the leading cold start path. This indicates that this node has started the network 0 No such event 1 <i>POC:normal active</i> reached from <i>POC:startup</i> state via leading cold start path
REMCSAT	Remaining Coldstart Attempts —protocol-related variable: <i>vRemainingColdstartAttempts</i> . This field provides the number of remaining cold start attempts that the CC will execute.
CPN	Leading Cold Start Path Noise —protocol-related variable: <i>vPOC!ColdstartNoise</i> . This status bit is set if the CC has reached the <i>POC:normal active</i> state via the leading cold start path under noise conditions. This indicates there was some activity on the FlexRay bus while the CC was starting up the cluster. 0 No such event 1 <i>POC:normal active</i> state was reached from <i>POC:startup</i> state via noisy leading cold start path
HHR	Host Halt Request Pending —protocol related variable: <i>vPOC!CHI!HaltRequest</i> . This status bit is set when CC receives the HALT command from the application via the <i>Protocol Operation Control Register (FR_POCR)</i> . The CC clears this status bit after a hard reset condition or when the protocol is in the <i>POC:default config</i> state. 0 No such event 1 HALT command received
FRZ	Freeze Occurred —protocol related variable: <i>vPOC!Freeze</i> . This status bit is set when the CC has reached the <i>POC:halt</i> state due to the host FREEZE command or due to an internal error condition requiring immediate halt. The CC clears this status bit after a hard reset condition or when the protocol is in the <i>POC:default config</i> state. 0 No such event 1 Immediate halt due to FREEZE or internal error condition
APTAC	Allow Passive to Active Counter —protocol-related variable: <i>vPOC!vAllowPassivetoActive</i> . This field provides the number of consecutive even/odd communication cycle pairs that have passed with valid rate and offset correction terms, but the protocol is still in the <i>POC:normal passive</i> state due to an application-configured delay to enter <i>POC:normal active</i> state. This delay is defined by the allow_passive_to_active field in <i>Protocol Configuration Register 12 (FR_PCR12)</i> .

51.5.2.23 Protocol Status Register 2 (FR_PSR2)

Figure 961. Protocol Status Register 2 (FR_PSR2)

Base + 0x002C				Additional Reset: RUN Command												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NBVB	NSEB	STCB	SBVB	SSEB	MTB	NBVA	NSEA	STCA	SBVA	SSEA	MTA	CLKCORRFAILCNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides a snapshot of status information about the network idle time (NIT), the symbol window, and the clock synchronization. The NIT-related status bits NBVB, NSEB, NBVA, and NSEA are updated by the CC after the end of the NIT and before the end of the first slot of the next communication cycle. The symbol-window-related status bits STCB, SBVB, SSEB, MTB, STCA, SBVA, SSEB, and MTA are updated by the CC after the end of the symbol window and before the end of the current communication cycle. If no symbol window is configured, the symbol-window-related status bits remain in their reset state. The clock synchronization-related CLKCORRFAILCNT is updated by the CC after the end of the static segment and before the end of the current communication cycle.

Table 996. FR_PSR2 field descriptions

Field	Description
NBVB	NIT Boundary Violation on Channel B —protocol-related variable: vSS!BViolation for NIT on channel B. This status bit is set when there was some media activity on the FlexRay bus channel B at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
NSEB	NIT Syntax Error on Channel B —protocol-related variable: vSS!SyntaxError for NIT on channel B. This status bit is set when a syntax error was detected during NIT on channel B. 0 No such event 1 Syntax error detected
STCB	Symbol Window Transmit Conflict on Channel B —protocol-related variable: vSS!TxConflict for symbol window on channel B. This status bit is set if there was a transmission conflict during the symbol window on channel B. 0 No such event 1 Transmission conflict detected
SBVB	Symbol Window Boundary Violation on Channel B —protocol-related variable: vSS!BViolation for symbol window on channel B. This status bit is set if there was some media activity on the FlexRay bus channel B at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
SSEB	Symbol Window Syntax Error on Channel B —protocol related variable: vSS!SyntaxError for symbol window on channel B. This status bit is set when a syntax error was detected during the symbol window on channel B. 0 No such event 1 Syntax error detected

Table 996. FR_PSR2 field descriptions (continued)

Field	Description
MTB	Media Access Test Symbol MTS Received on Channel B —protocol-related variable: vSS!ValidMTS for Symbol Window on channel B. This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel B. 0 No such event 1 MTS symbol received
NBVA	NIT Boundary Violation on Channel A —protocol-related variable: vSS!BViolation for NIT on channel A. This status bit is set when there was some media activity on the FlexRay bus channel A at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
NSEA	NIT Syntax Error on Channel A —protocol-related variable: vSS!SyntaxError for NIT on channel A. This status bit is set when a syntax error was detected during NIT on channel A. 0 No such event 1 Syntax error detected
STCA	Symbol Window Transmit Conflict on Channel A —protocol related variable: vSS!TxConflict for symbol window on channel A. This status bit is set if there was a transmission conflicts during the symbol window on channel A. 0 No such event 1 Transmission conflict detected
SBVA	Symbol Window Boundary Violation on Channel A —protocol-related variable: vSS!BViolation for symbol window on channel A. This status bit is set if there was some media activity on the FlexRay bus channel A at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
SSEA	Symbol Window Syntax Error on Channel A —protocol-related variable: vSS!SyntaxError for symbol window on channel A. This status bit is set when a syntax error was detected during the symbol window on channel A. 0 No such event 1 Syntax error detected
MTA	Media Access Test Symbol MTS Received on Channel A —protocol-related variable: vSS!ValidMTS for symbol window on channel A. This status bit is set if the Media Access Test Symbol (MTS) was received in the symbol window on channel A. 0 No such event 1 MTS symbol received
CLKCORR-FAILCNT	Clock Correction Failed Counter —protocol -elated variable: vClockCorrectionFailed . This field provides the number of consecutive even/odd communication cycle pairs that have passed without a clock synchronization offset or rate correction due to lack of synchronization frames. It is not incremented when it has reached the configured value of either <code>max_without_clock_correction_fatal</code> or <code>max_without_clock_correction_passive</code> as defined in Protocol Configuration Register 8 (FR_PCR8) . The CC resets this counter on a hard reset condition, when the protocol enters the POC:normal active state, or when both the rate and offset correction terms have been calculated successfully.

51.5.2.24 Protocol Status Register 3 (FR_PSR3)

Figure 962. Protocol Status Register 3 (FR_PSR3)

Base + 0x002E				Additional Reset: RUN Command								Write: Normal mode				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	WUB	ABVB	AACB	ACEB	ASEB	AVFB	0	0	WUA	ABVA	AACA	ACEA	ASEA	AVFA
W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides aggregated channel status information as an accrued status of channel activity for all communication slots, regardless of whether they are assigned for transmission or subscribed for reception. It provides accrued information for the symbol window, the NIT, and the wakeup status.

Table 997. FR_PSR3 field descriptions

Field	Description
WUB	Wakeup Symbol Received on Channel B —This flag is set when a wakeup symbol was received on channel B. 0 No wakeup symbol received 1 Wakeup symbol received
ABVB	Aggregated Boundary Violation on Channel B —This flag is set when a boundary violation was detected on channel B. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
AACB	Aggregated Additional Communication on Channel B —This flag is set when at least one valid frame was received on channel B in a slot that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
ACEB	Aggregated Content Error on Channel B —This flag is set when a content error was detected on channel B. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
ASEB	Aggregated Syntax Error on Channel B —This flag is set when a syntax error was detected on channel B. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
AVFB	Aggregated Valid Frame on Channel B —This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel B. 0 No syntactically valid frames received 1 At least one syntactically valid frame received
WUA	Wakeup Symbol Received on Channel A —This flag is set when a wakeup symbol was received on channel A. 0 No wakeup symbol received 1 Wakeup symbol received

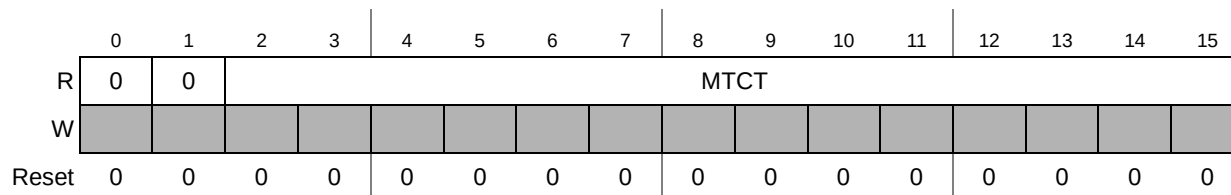
Table 997. FR_PSR3 field descriptions (continued)

Field	Description
ABVA	Aggregated Boundary Violation on Channel A —This flag is set when a boundary violation was detected on channel A. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
AACA	Aggregated Additional Communication on Channel A —This flag is set when a valid frame was received in a slot on channel A that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
ACEA	Aggregated Content Error on Channel A —This flag is set when a content error has been detected on channel A. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
ASEA	Aggregated Syntax Error on Channel A —This flag is set when a syntax error has been detected on channel A. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
AVFA	Aggregated Valid Frame on Channel A —This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel A. 0 No syntactically valid frames received 1 At least one syntactically valid frame received

51.5.2.25 MacroTICK Counter Register (FR_MTCTR)

Figure 963. MacroTICK Counter Register (FR_MTCTR)

Base + 0x0030



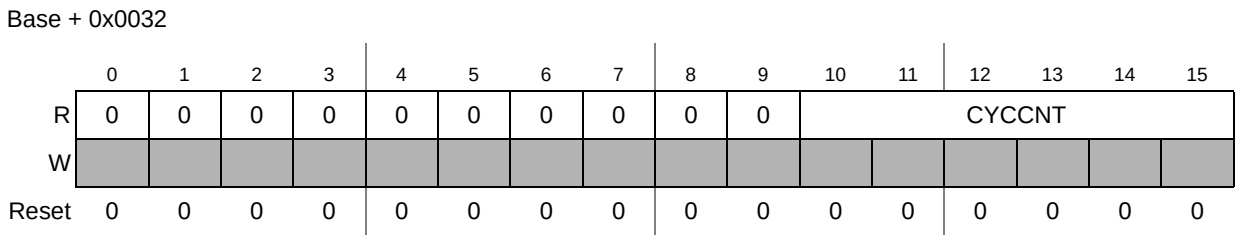
This register provides the macroTICK count of the current communication cycle.

Table 998. FR_MTCTR field descriptions

Field	Description
MTCT	MacroTICK Counter —protocol-related variable: vMacroTICK . This field provides the macroTICK count of the current communication cycle.

51.5.2.26 Cycle Counter Register (FR_CYCTR)

Figure 964. Cycle Counter Register (FR_CYCTR)



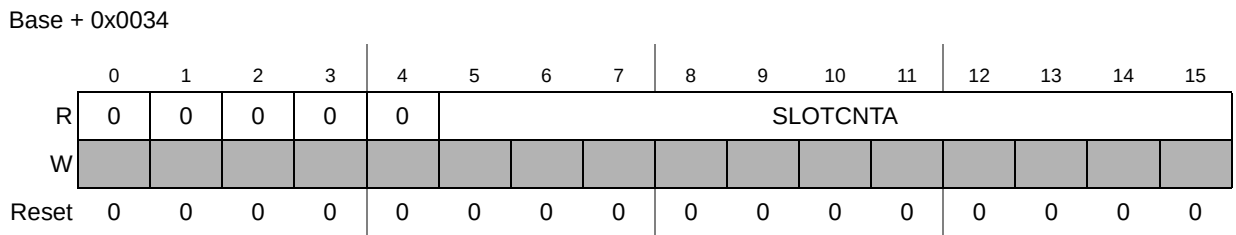
This register provides the number of the current communication cycle.

Table 999. FR_CYCTR field descriptions

Field	Description
CYCCNT	Cycle Counter —protocol-related variable: vCycleCounter . This field provides the number of the current communication cycle. If the counter reaches the maximum value of 63, the counter wraps and starts from 0 again.

51.5.2.27 Slot Counter Channel A Register (FR_SLTCTAR)

Figure 965. Slot Counter Channel A Register (FR_SLTCTAR)



This register provides the number of the current slot in the current communication cycle for channel A.

Table 1000. FR_SLTCTAR field descriptions

Field	Description
SLOTCNTA	Slot Counter Value for Channel A —protocol-related variable: vSlotCounter for channel A. This field provides the number of the current slot in the current communication cycle.

51.5.2.28 Slot Counter Channel B Register (FR_SLTCTBR)

Figure 966. Slot Counter Channel B Register (FR_SLTCTBR)

Base + 0x0036

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	SLOTCNTB										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides the number of the current slot in the current communication cycle for channel B.

Table 1001. FR_SLTCTBR field descriptions

Field	Description
SLOTCNTA	Slot Counter Value for Channel B —protocol-related variable: <i>vSlotCounter</i> for channel B. This field provides the number of the current slot in the current communication cycle.

51.5.2.29 Rate Correction Value Register (FR_RTCORVR)

Figure 967. Rate Correction Value Register (FR_RTCORVR)

Base + 0x0038

Additional Reset: RUN Command

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RATECORR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

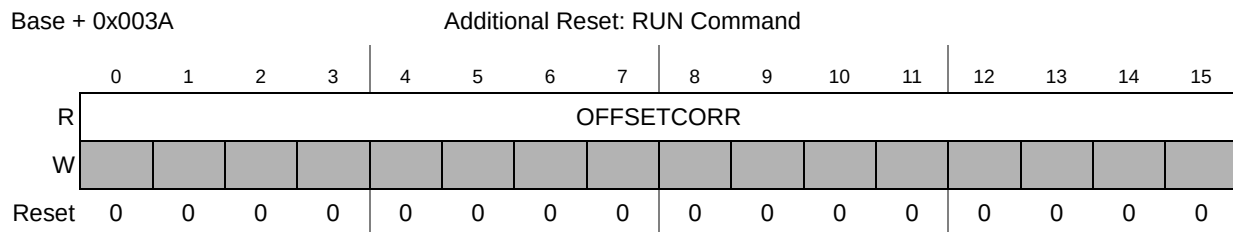
This register provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT of each odd numbered communication cycle.

Table 1002. FR_RTCORVR Field Descriptions

Field	Description
RATECORR	Rate Correction Value —protocol-related variable: <i>vRateCorrection</i> (before value limitation and external rate correction). This field provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external rate correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by <i>rate_correction_out</i> in the Protocol Configuration Register 13 (FR_PCR13) , the clock correction reached limit interrupt flag (CCL_IF) is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0) . <i>Note: If the CC was not able to calculate a new rate correction term due to a lack of synchronization frames, the RATECORR value is not updated.</i>

51.5.2.30 Offset Correction Value Register (FR_OFCORVR)

Figure 968. Offset Correction Value Register (FR_OFCORVR)



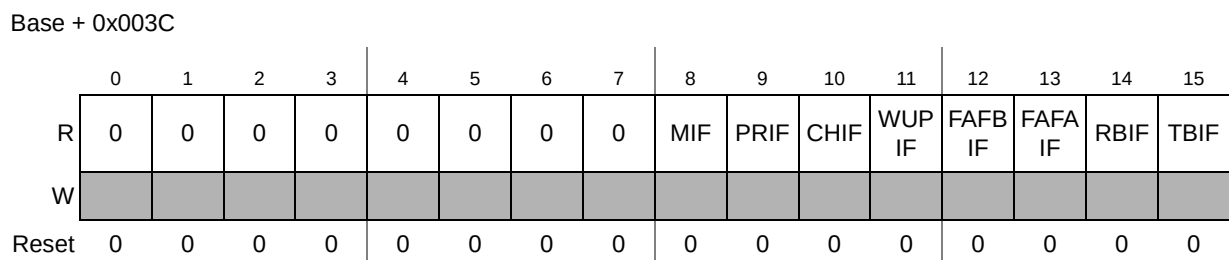
This register provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT.

Table 1003. FR_OFCORVR Field Descriptions

Field	Description
OFFSET-CORR	Offset Correction Value —protocol-related variable: <i>vOffsetCorrection</i> (before value limitation and external offset correction). This field provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external offset correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by <code>offset_correction_out</code> field in the Protocol Configuration Register 29 (FR_PCR29) , the clock correction reached limit interrupt flag <code>CCL_IF</code> is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0) . <i>Note: If the CC was not able to calculate an new offset correction term due to a lack of synchronization frames, the OFFSETCORR value is not updated.</i>

51.5.2.31 Combined Interrupt Flag Register (FR_CIFR)

Figure 969. Combined Interrupt Flag Register (FR_CIFR)



This register provides five combined interrupt flags and a copy of three individual interrupt flags. The combined interrupt flags are the result of a binary OR of the values of other interrupt flags regardless of the state of the interrupt enable bits. The generation scheme for the combined interrupt flags is depicted in [Figure 1100](#). The individual interrupt flags `WUPIF`, `FAFBIF`, and `FAFAIF` are copies of corresponding flags in the [Global Interrupt Flag and Enable Register \(FR_GIFER\)](#) and are provided here to simplify the application interrupt flag check. To clear the individual interrupt flags, the application must use the [Global Interrupt Flag and Enable Register \(FR_GIFER\)](#).

Note: The meanings of the combined status bits MIF, PRIF, CHIF, RBIF, and TBIF are different from those mentioned in the [Global Interrupt Flag and Enable Register \(FR_GIFER\)](#).

Table 1004. FR_CIFR field descriptions

Field	Description
MIF	Module Interrupt Flag —This flag is set if there is at least one interrupt source that has its interrupt flag asserted. 0 No interrupt source has its interrupt flag asserted 1 At least one interrupt source has its interrupt flag asserted
PRIF	Protocol Interrupt Flag —This flag is set if at least one of the individual protocol interrupt flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) or Protocol Interrupt Flag Register 1 (FR_PIFR1) is equal to 1. 0 All individual protocol interrupt flags are equal to 0 1 At least one of the individual protocol interrupt flags is equal to 1
CHIF	CHI Interrupt Flag —This flag is set if at least one of the individual CHI error flags in the CHI Error Flag Register (FR_CHIERFR) is equal to 1. 0 All CHI error flags are equal to 0 1 At least one CHI error flag is equal to 1
WUPIF	Wakeup Interrupt Flag —Provides the same value as FR_GIFER[WUPIF].
FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag —Provides the same value as FR_GIFER[FAFBIF].
FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag —Provides the same value as FR_GIFER[FAFAIF].
RBIF	Receive Message Buffer Interrupt Flag —This flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual receive message buffers has the MBIF flag asserted 1 At least one individual receive message buffers has the MBIF flag asserted
TBIF	Transmit Message Buffer Interrupt Flag —This flag is set if for at least one of the individual transmit message buffers (FR_MBCCSRn[MTD] = 1) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual transmit message buffers has the MBIF flag asserted 1 At least one individual transmit message buffers has the MBIF flag asserted

51.5.2.32 System Memory Access Timeout Register (FR_SYMATOR)

Figure 970. System Memory Access Time-Out Register (FR_SYMATOR)

Base + 0x003E								Write: Disabled mode								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TIMEOUT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Table 1005. FR_SYMATOR field descriptions

Field	Description
TIMEOUT	System Memory Access Timeout —This value defines when a system bus access timeout is detected. For a detailed description see Section 51.7.1.1: Configure System Memory Access Timeout Register (FR_SYMATOR) and Section 51.6.19.1.2: System bus access timeout .

51.5.2.33 Sync Frame Counter Register (FR_SFCNTR)

Figure 971. Sync Frame Counter Register (FR_SFCNTR)

Base + 0x0040				Additional Reset: RUN Command												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SFEVB				SFEVA				SFODB				SFODA			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides the number of synchronization frames that are used for clock synchronization in the last even and in the last odd numbered communication cycle. This register is updated after the start of the NIT and before 10 macroticks after offset correction start.

Note: *If the application has locked the even synchronization table at the end of the static segment of an even communication cycle, the CC will not update the fields SFEVB and SFEVA.*
If the application has locked the odd synchronization table at the end of the static segment of an odd communication cycle, the CC will not update the SFODB and SFODA values.

Table 1006. FR_SFCNTR Field Descriptions

Field	Description
SFEVB	Sync Frames Channel B, even cycle—protocol-related variable: size of (vsSyncIdListB for even cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
SFEVA	Sync Frames Channel A, even cycle—protocol related variable: size of (vsSyncIdListA for even cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
SFODB	Sync Frames Channel B, odd cycle—protocol-related variable: size of (vsSyncIdListB for odd cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
SFODA	Sync Frames Channel A, odd cycle—protocol-related variable: size of (vsSyncIdListA for odd cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.

51.5.2.34 Sync Frame Table Offset Register (FR_SFTOR)

Figure 972. Sync Frame Table Offset Register (FR_SFTOR)

Base + 0x0042

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SFT_OFFSET[15:1]															0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines the FlexRay memory area related offset for sync frame tables. For more details, see [Section 51.6.12, “Sync frame ID and sync frame deviation tables”](#).

Table 1007. FR_SFTOR field description

Field	Description
SFT_OFFSET	Sync Frame Table Offset —The offset of the sync frame tables in the FlexRay memory area. This offset is required to be 16-bit aligned. Thus, STF_OFFSET[0] is always 0.

51.5.2.35 Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

Figure 973. Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

Base + 0x0044

Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	CYCNUM						ELKS	OLKS	EVAL	OVAL	0	0	SDV EN	SID EN
W	ELKT	OLKT														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides configuration, control, and status information related to the generation and access of the clock sync ID tables and clock sync measurement tables. For a detailed description, see [Section 51.6.12, “Sync frame ID and sync frame deviation tables”](#).

Table 1008. FR_SFTCCSR field descriptions

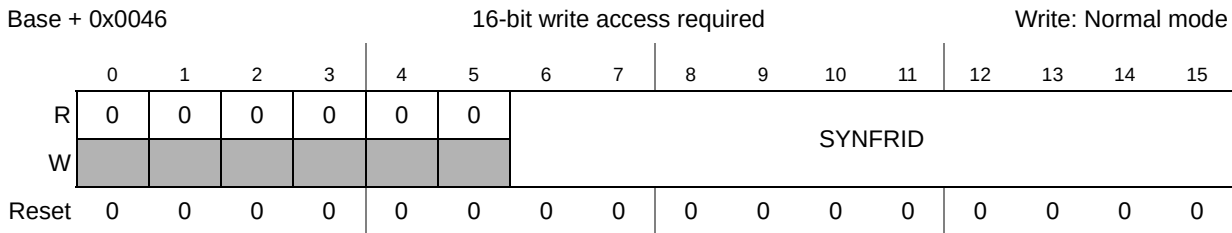
Field	Description
ELKT	Even Cycle Tables Lock/Unlock Trigger —This trigger bit is used to lock and unlock the even cycle tables. 0 No effect 1 Triggers lock/unlock of the even cycle tables
OLKT	Odd Cycle Tables Lock/Unlock Trigger —This trigger bit is used to lock and unlock the odd cycle tables. 0 No effect 1 Triggers lock/unlock of the odd cycle tables
CYCNUM	Cycle Number —This field provides the number of the cycle in which the currently locked table was recorded. If none or both tables are locked, this value is related to the even cycle table.

Table 1008. FR_SFTCCSR field descriptions

Field	Description
ELKS	Even Cycle Tables Lock Status —This status bit indicates whether the application has locked the even cycle tables. 0 Application has not locked the even cycle tables 1 Application has locked the even cycle tables
OLKS	Odd Cycle Tables Lock Status —This status bit indicates whether the application has locked the odd cycle tables. 0 Application has not locked the odd cycle tables 1 Application has locked the odd cycle tables
EVAL	Even Cycle Tables Valid —This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the even cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent)
OVAL	Odd Cycle Tables Valid —This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the odd cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent)
OPT	One Pair Trigger —This trigger bit controls whether the CC writes continuously or only one pair of Sync Frame Tables into the FlexRay memory area. If this trigger is set to 1 while SDVEN or SIDEN is set to 1, the CC writes only one pair of the enabled Sync Frame Tables corresponding to the next even-odd-cycle pair into the FlexRay memory area. In this case, the CC clears the SDVEN or SIDEN bits immediately. If this trigger is set to 0 while SDVEN or SIDEN is set to 1, the CC writes continuously the enabled Sync Frame Tables into the FlexRay memory area. 0 Write continuously pairs of enabled Sync Frame Tables into FlexRay memory area. 1 Write only one pair of enabled Sync Frame Tables into FlexRay memory area.
SDVEN	Sync Frame Deviation Table Enable —This bit controls the generation of the Sync Frame Deviation Tables. The application must set this bit to request the CC to write the Sync Frame Deviation Tables into the FlexRay memory area. 0 Do not write Sync Frame Deviation Tables 1 Write Sync Frame Deviation Tables into FlexRay memory area <i>Note: If SDVEN is set to 1, then SIDEN must also be set to 1.</i>
SIDEN	Sync Frame ID Table Enable —This bit controls the generation of the Sync Frame ID Tables. The application must set this bit to 1 to request the CC to write the Sync Frame ID Tables into the FlexRay memory area. 0 Do not write Sync Frame ID Tables 1 Write Sync Frame ID Tables into FlexRay memory area

51.5.2.36 Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Figure 974. Sync Frame ID Rejection Filter Register (FR_SFIDRFR)



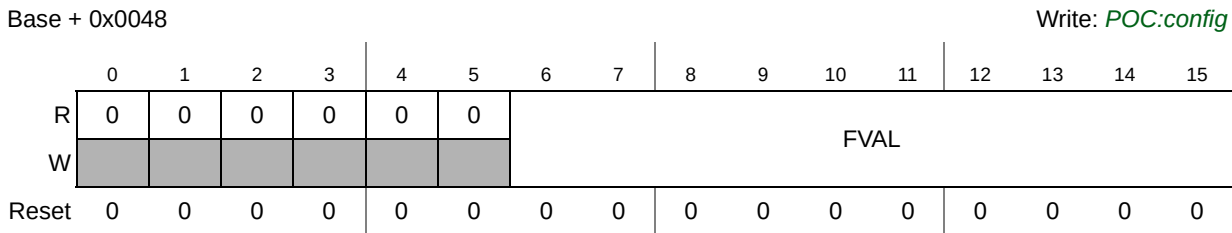
This register defines the Sync Frame Rejection Filter ID. The application must update this register outside of the static segment. If the application updates this register in the static segment, it can appear that the CC accepts the sync frame in the current cycle.

Table 1009. FR_SFIDRFR field descriptions

Field	Description
SYNFRID	Sync Frame Rejection ID —This field defines the frame ID of a frame that must not be used for clock synchronization. For details see Section 51.6.15.2, “Sync Frame Rejection Filtering .

51.5.2.37 Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

Figure 975. Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)



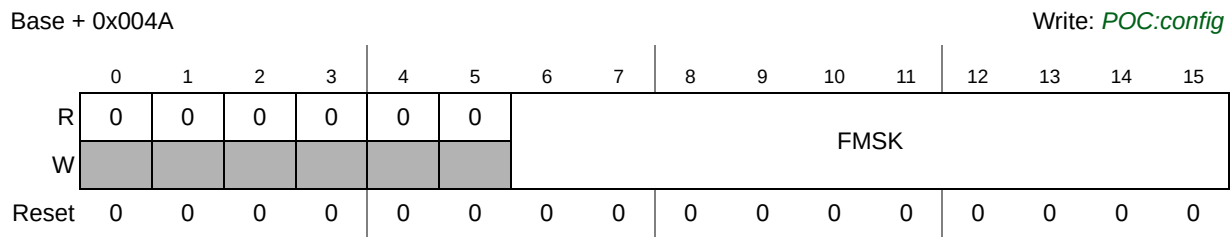
This register defines the sync frame acceptance filter value. For details on filtering, see [Section 51.6.15, “Sync frame filtering](#).

Table 1010. FR_SFIDAFVR Field Descriptions

Field	Description
FVAL	Filter Value —This field defines the value for the sync frame acceptance filtering.

51.5.2.38 Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

Figure 976. Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)



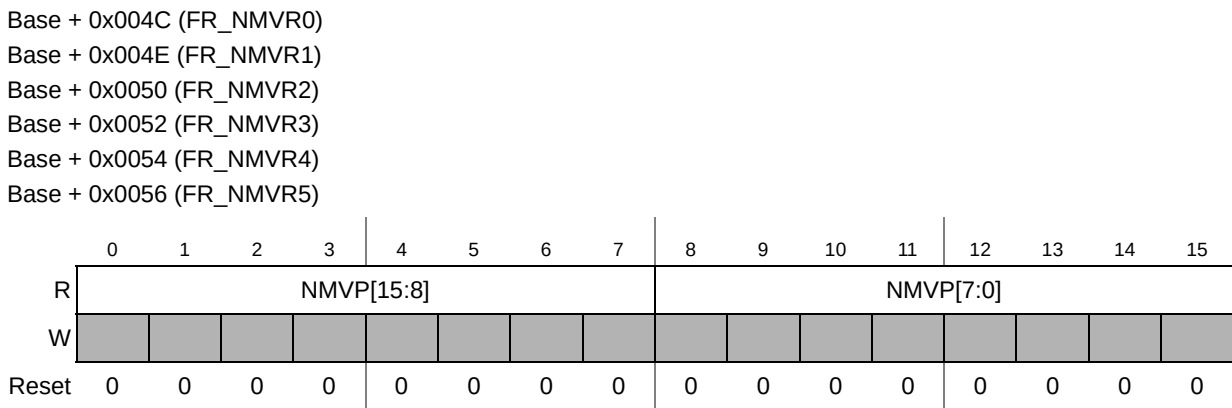
This register defines the sync frame acceptance filter mask. For details on filtering see [Section 51.6.15.1, “Sync frame acceptance filtering”](#).

Table 1011. FR_SFIDAFMR Field Descriptions

Field	Description
FMSK	Filter Mask —This field defines the mask for the sync frame acceptance filtering.

51.5.2.39 Network Management Vector Registers (FR_NMVR0–FR_NMVR5)

Figure 977. Network Management Vector Registers (FR_NMVR0–FR_NMVR5)



Each of these six registers holds one part of the Network Management Vector. The length of the Network Management Vector is configured in the [Network Management Vector Length Register \(FR_NMVLR\)](#). If FR_NMVLR is programmed with a value that is less than 12 bytes, the remaining bytes of the [Network Management Vector Registers \(FR_NMVR0–FR_NMVR5\)](#), which are not used for the Network Management Vector accumulating, will remain 0.

The NMVR provides accrued information over all received NMVs in the last communication cycle. All NMVs received in one cycle are ORed into the NMVR. The NMVR is updated at the end of the communication cycle.



Table 1012. NMVR[0:5] field descriptions

Field	Description
NMVP	Network Management Vector Part —The mapping between the <i>Network Management Vector Registers</i> (<i>FR_NMVR0–FR_NMVR5</i>) and the receive message buffer payload bytes in NMV[0:11] is depicted in <i>Table 1013</i> .

Table 1013. Mapping of NMVRn to the Received Payload Bytes NMVn

NMVRn Register	NMVn Received Payload
FR_NMVR0[NMVP[15:8]]	NMV0
FR_NMVR0[NMVP[7:0]]	NMV1
FR_NMVR1[NMVP[15:8]]	NMV2
FR_NMVR1[NMVP[7:0]]	NMV3
...	
FR_NMVR5[NMVP[15:8]]	NMV10
FR_NMVR5[NMVP[7:0]]	NMV11

51.5.2.40 Network Management Vector Length Register (FR_NMVLRL)

Figure 978. Network Management Vector Length Register (FR_NMVLRL)

Base + 0x0058

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	NMVL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines the length of the network management vector in bytes.

Table 1014. FR_NMVLRL field descriptions

Field	Description
NMVL	Network Management Vector Length —protocol-related variable: <i>gNetworkManagementVectorLength</i> . This field defines the length of the Network Management Vector in bytes. Legal values are between 0 and 12.

51.5.2.41 Timer Configuration and Control Register (FR_TICCR)

Figure 979. Timer Configuration and Control Register (FR_TICCR)

Base + 0x005A

Write: T2_CFG: *POC:config*
T2_REP, T1_REP, T1SP, T2SP, T1TR, T2TR: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	T2_CFG	T2_REP	0	0	0	T2ST	0	0	0	T1_REP	0	0	0	T1ST
W						T2SP	T2TR							T1SP	T1TR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register is used to configure and control the T1 and T2 timers. For timer details, see [Section 51.6.17, “Timer support”](#). The T1 timer is an absolute timer. The T2 timer can be configured as an absolute or relative timer.

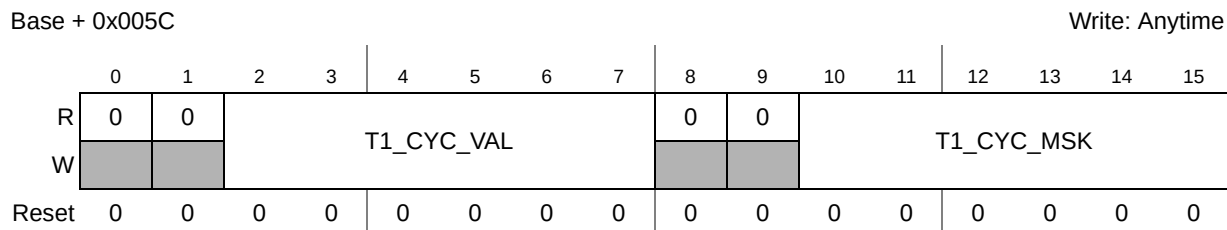
Table 1015. FR_TICCR field descriptions

Field	Description
T2_CFG	Timer T2 Configuration —This bit configures the timebase mode of timer T2. 0 T2 is absolute timer. 1 T2 is relative timer.
T2_REP	Timer T2 Repetitive Mode —This bit configures the repetition mode of timer T2. 0 T2 is non repetitive 1 T2 is repetitive
T2SP	Timer T2 Stop —This trigger bit is used to stop timer T2. 0 No effect 1 Stop timer T2
T2TR	Timer T2 Trigger —This trigger bit is used to start timer T2. 0 No effect 1 Start timer T2
T2ST	Timer T2 State —This status bit provides the current state of timer T2. 0 Timer T2 is idle 1 Timer T2 is running
T1_REP	Timer T1 Repetitive Mode —This bit configures the repetition mode of timer T1. 0 T1 is non repetitive 1 T1 is repetitive
T1SP	Timer T1 Stop —This trigger bit is used to stop timer T1. 0 No effect 1 Stop timer T1
T1TR	Timer T1 Trigger —This trigger bit is used to start timer T1. 0 No effect 1 Start timer T1
T1ST	Timer T1 State —This status bit provides the current state of timer T1. 0 Timer T1 is idle 1 Timer T1 is running

Note: Both timers are deactivated immediately when the protocol enters a state different from *POC:normal active* or *POC:normal passive*.

51.5.2.42 Timer 1 Cycle Set Register (FR_TI1CYSR)

Figure 980. Timer 1 Cycle Set Register (FR_TI1CYSR)



This register defines the cycle filter value and the cycle filter mask for timer T1. For a detailed description of timer T1, refer to [Section 51.6.17.1, “Absolute timer T1”](#).

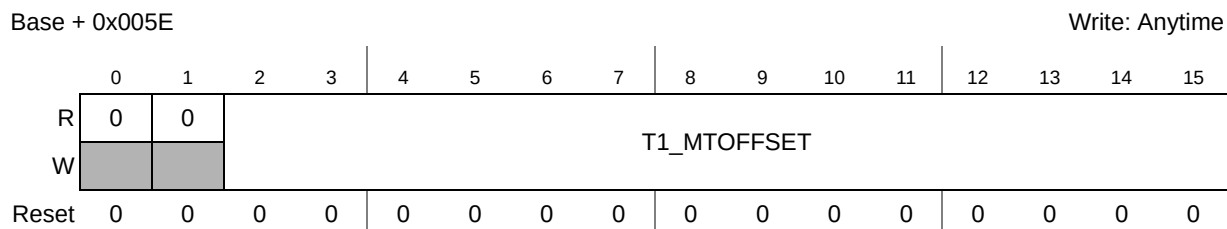
Table 1016. FR_TI1CYSR Field Descriptions

Field	Description
T1_CYC_VAL	Timer T1 Cycle Filter Value —This field defines the cycle filter value for timer T1.
T1_CYC_MSK	Timer T1 Cycle Filter Mask —This field defines the cycle filter mask for timer T1.

Note: If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

51.5.2.43 Timer 1 Macrotick Offset Register (FR_TI1MTOR)

Figure 981. Timer 1 Macrotick Offset Register (FR_TI1MTOR)



This register holds the macrotick offset value for timer T1. For a detailed description of timer T1, refer to [Section 51.6.17.1, “Absolute timer T1”](#).

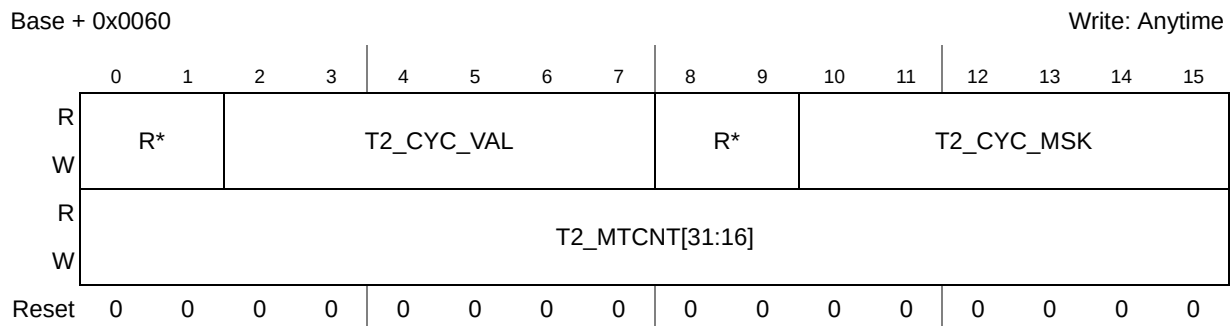
Table 1017. FR_TI1MTOR Field Descriptions

Field	Description
T1_MTOFFSET	Timer 1 Macrotick Offset —This field defines the macrotick offset value for timer 1.

Note: If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

51.5.2.44 Timer 2 Configuration Register 0 (FR_TI2CR0)

Figure 982. Timer 2 Configuration Register 0 (FR_TI2CR0)



The content of this register depends on the value of the T2_CFG bit in the [Timer Configuration and Control Register \(FR_TICCR\)](#). For a detailed description of timer T2, refer to [Section 51.6.17.2, "Absolute/relative timer T2"](#).

Table 1018. FR_TI2CR0 field descriptions

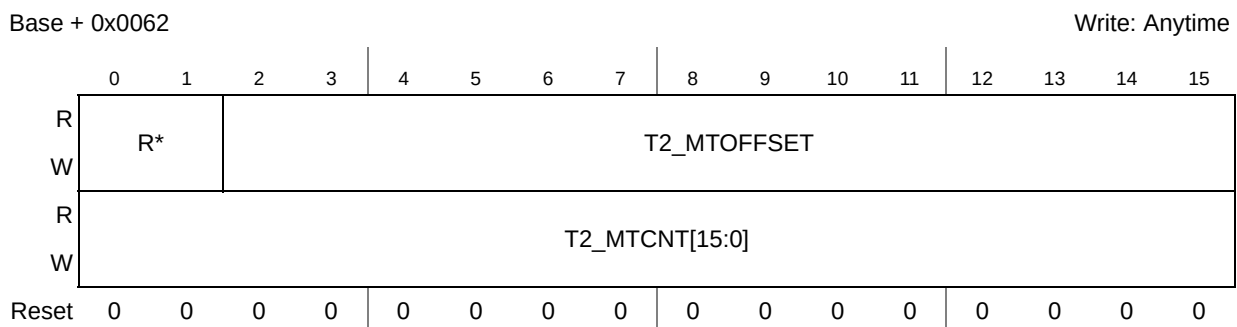
Field	Description
Fields for absolute timer T2 (FR_TICCR[T2_CFG] = 0)	
T2_CYC_VAL	Timer T2 Cycle Filter Value —This field defines the cycle filter value for timer T2.
T2_CYC_MSK	Timer T2 Cycle Filter Mask —This field defines the cycle filter mask for timer T2.
Fields for relative timer T2 (FR_TICCR[T2_CFG] = 1)	
T2_MTCNT[31:16]	Timer T2 Macrotick High Word —This field defines the high word of the macrotick count for timer T2.

Note: *If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and timer T2 will expire according to the changed values.*

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

51.5.2.45 Timer 2 Configuration Register 1 (FR_TI2CR1)

Figure 983. Timer 2 Configuration Register 1 (FR_TI2CR1)



The content of this register depends on the value of the T2_CFG bit in the [Timer Configuration and Control Register \(FR_TICCR\)](#). For a detailed description of timer T2, refer to [Section 51.6.17.2, “Absolute/relative timer T2](#).

Table 1019. FR_TI2CR1 field descriptions

Field	Description
Fields for absolute timer T2 (FR_TICCR[T2_CFG] = 0)	
T2_MTOFFSET	Timer T2 Macrotick Offset —This field holds the macrotick offset value for timer T2.
Fields for relative timer T2 (FR_TICCR[T2_CFG] = 1)	
T2_MTCNT[15:0]	Timer T2 Macrotick Low Word —This field defines the low word of the macrotick value for timer T2.

Note: *If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and the timer T2 will expire according to the changed values.*

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

51.5.2.46 Slot Status Selection Register (FR_SSSR)

Figure 984. Slot Status Selection Register (FR_SSSR)

Base + 0x0064				16-bit write access required								Write: Anytime				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SEL		0	SLOTNUMBER										
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register is used to access the four internal non-memory-mapped Slot Status Selection Registers FR_SSSR0 to FR_SSSR3. Each internal register selects a slot, or symbol window/NIT, whose status vector will be saved in the corresponding [Slot Status Registers \(FR_SSR0–FR_SSR7\)](#) according to [Table 1021](#). For a detailed description of slot status monitoring, refer to [Section 51.6.18, “Slot status monitoring](#).

Table 1020. FR_SSSR field descriptions

Field	Description
WMD	Write Mode —This control bit defines the write mode of this register. 0 Write to all fields in this register on write access 1 Write to SEL field only on write access

Table 1020. FR_SSSR field descriptions

Field	Description
SEL	Selector —This field selects one of the four internal slot status selection registers for access. 00 Select FR_SSSR0 01 Select FR_SSSR1 10 Select FR_SSSR2 11 Select FR_SSSR3
SLOTNUMBER	Slot Number —This field specifies the number of the slot whose status will be saved in the corresponding slot status registers. <i>Note: If this value is set to 0, the related slot status register provides the status of the symbol window after the NIT start, and provides the status of the NIT after the cycle start.</i>

Table 1021. Mapping between FR_SSSRn and FR_SSRn

Internal Slot Status Selection Register	Write the Slot Status of the Slot Selected by FR_SSSRn for each			
	Even Communication Cycle		Odd Communication Cycle	
	For Channel B to	For Channel A to	For Channel B to	For Channel A to
FR_SSSR0	FR_SSR0[15:8]	FR_SSR0[7:0]	FR_SSR1[15:8]	FR_SSR1[7:0]
FR_SSSR1	FR_SSR2[15:8]	FR_SSR2[7:0]	FR_SSR3[15:8]	FR_SSR3[7:0]
FR_SSSR2	FR_SSR4[15:8]	FR_SSR4[7:0]	FR_SSR5[15:8]	FR_SSR5[7:0]
FR_SSSR3	FR_SSR6[15:8]	FR_SSR6[7:0]	FR_SSR7[15:8]	FR_SSR7[7:0]

51.5.2.47 Slot Status Counter Condition Register (FR_SSCCR)

Figure 985. Slot Status Counter Condition Register (FR_SSCCR)

Base + 0x0066				16-bit write access required								Write: Anytime				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SEL		0	CNTCFG		MCY	VFR	SYF	NUF	SUF	STATUSMASK[3:0]			
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register is used to access and program the four internal non-memory-mapped Slot Status Counter Condition Registers FR_SSCCR0 to FR_SSCCR3. Each of these four internal Slot Status Counter Condition Registers defines the mode and the conditions for incrementing the counter in the corresponding [Slot Status Counter Registers \(FR_SSCR0–FR_SSCR3\)](#). The correspondence is given in [Table 1023](#). For a detailed description of slot status counters, refer to [Section 51.6.18.4, “Slot Status Counter Registers](#).

Table 1022. FR_SSCCR field descriptions

Field	Description
WMD	Write Mode —This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
SEL	Selector —This field selects one of the four internal slot counter condition registers for access. 00 select FR_SSCCR0 01 select FR_SSCCR1 10 select FR_SSCCR2 11 select FR_SSCCR3
CNTCFG	Counter Configuration —These bit field controls the channel related incrementing of the slot status counter. 00 Increment by 1 if condition is fulfilled on channel A. 01 Increment by 1 if condition is fulfilled on channel B. 10 Increment by 1 if condition is fulfilled on at least one channel. 11 Increment by 2 if condition is fulfilled on both channels channel. Increment by 1 if condition is fulfilled on only one channel.
MCY	Multi Cycle Selection —This bit defines whether the slot status counter accumulates over multiple communication cycles or provides information for the previous communication cycle only. 0 The Slot Status Counter provides information for the previous communication cycle only 1 The Slot Status Counter accumulates over multiple communication cycles
VFR	Valid Frame Restriction —This bit is used to restrict the counter to received valid frames. 0 The counter is not restricted to valid frames only 1 The counter is restricted to valid frames only
SYF	Sync Frame Restriction —This bit is used to restrict the counter to received frames with the sync frame indicator bit set to 1. 0 The counter is not restricted with respect to the sync frame indicator bit 1 The counter is restricted to frames with the sync frame indicator bit set to 1
NUF	Null Frame Restriction —This bit is used to restrict the counter to received frames with the null frame indicator bit set to 0. 0 The counter is not restricted with respect to the null frame indicator bit 1 The counter is restricted to frames with the null frame indicator bit set to 0
SUF	Startup Frame Restriction —This bit is used to restrict the counter to received frames with the startup frame indicator bit set to 1. 0 The counter is not restricted with respect to the startup frame indicator bit 1 The counter is restricted to received frames with the startup frame indicator bit set to 1
STATUS MASK[3:0]	Slot Status Mask —This bit field is used to enable the counter with respect to the four slot status error indicator bits. STATUSMASK[3] —This bit enables the counting for slots with the syntax error indicator bit set to 1 STATUSMASK[2] —This bit enables the counting for slots with the content error indicator bit set to 1 STATUSMASK[1] —This bit enables the counting for slots with the boundary violation indicator bit set to 1 STATUSMASK[0] —This bit enables the counting for slots with the transmission conflict indicator bit set to 1

Table 1023. Mapping between internal FR_SSCCRn and FR_SSCRn

Condition Register	Condition Defined for Register
FR_SSCCR0	FR_SSCR0
FR_SSCCR1	FR_SSCR1
FR_SSCCR2	FR_SSCR2
FR_SSCCR3	FR_SSCR3

51.5.2.48 Slot Status Registers (FR_SSR0–FR_SSR7)

Figure 986. Slot Status Registers (FR_SSR0–FR_SSR7)

Base + 0x0068 (FR_SSR0)

Base + 0x006A (FR_SSR1)

Base + 0x006C (FR_SSR2)

Base + 0x006E (FR_SSR3)

Base + 0x0070 (FR_SSR4)

Base + 0x0072 (FR_SSR5)

Base + 0x0074 (FR_SSR6)

Base + 0x0076 (FR_SSR7)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each of these eight registers holds the status vector of the slot specified in the corresponding internal Slot Status Selection Register, which can be programmed using the [Slot Status Selection Register \(FR_SSSR\)](#). Each register is updated after the end of the corresponding slot as shown in [Figure 1096](#). The register bits are directly related to the protocol variables and described in more detail in [Section 51.6.18, “Slot status monitoring”](#).

Table 1024. FR_SSR0–FR_SSR7 field descriptions

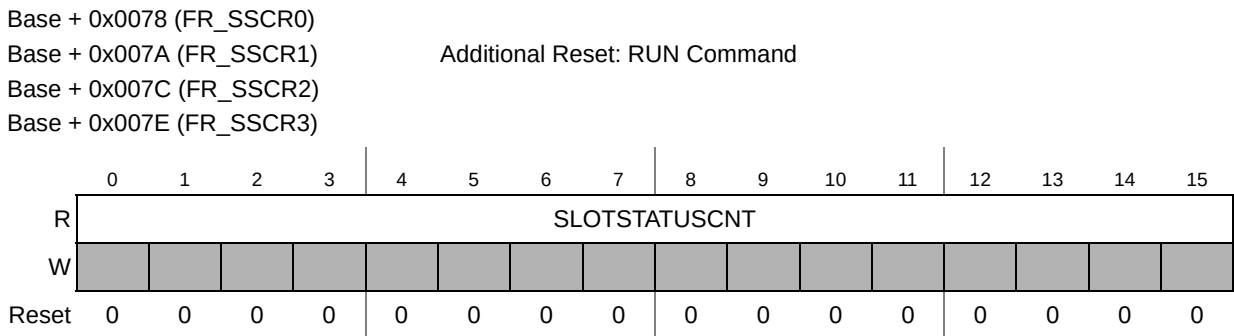
Field	Description
VFB	Valid Frame on Channel B —protocol-related variable: vSS!ValidFrame channel B. 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
SYB	Sync Frame Indicator Channel B —protocol-related variable: vRF!Header!SyFIndicator channel B. 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
NFB	Null Frame Indicator Channel B —protocol-related variable: vRF!Header!NFIndicator channel B. 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1

Table 1024. FR_SSR0–FR_SSR7 field descriptions

Field	Description
SUB	Startup Frame Indicator Channel B —protocol-related variable: <i>vRF!Header!SuFIndicator</i> channel B. 0 <i>vRF!Header!SuFIndicator</i> = 0 1 <i>vRF!Header!SuFIndicator</i> = 1
SEB	Syntax Error on Channel B —protocol-related variable: <i>vSS!SyntaxError</i> channel B. 0 <i>vSS!SyntaxError</i> = 0 1 <i>vSS!SyntaxError</i> = 1
CEB	Content Error on Channel B —protocol-related variable: <i>vSS!ContentError</i> channel B. 0 <i>vSS!ContentError</i> = 0 1 <i>vSS!ContentError</i> = 1
BVB	Boundary Violation on Channel B —protocol-related variable: <i>vSS!BViolation</i> channel B. 0 <i>vSS!BViolation</i> = 0 1 <i>vSS!BViolation</i> = 1
TCB	Transmission Conflict on Channel B —protocol-related variable: <i>vSS!TxConflict</i> channel B. 0 <i>vSS!TxConflict</i> = 0 1 <i>vSS!TxConflict</i> = 1
VFA	Valid Frame on Channel A —protocol-related variable: <i>vSS!ValidFrame</i> channel A. 0 <i>vSS!ValidFrame</i> = 0 1 <i>vSS!ValidFrame</i> = 1
SYA	Sync Frame Indicator Channel A —protocol-related variable: <i>vRF!Header!SyFIndicator</i> channel A. 0 <i>vRF!Header!SyFIndicator</i> = 0 1 <i>vRF!Header!SyFIndicator</i> = 1
NFA	Null Frame Indicator Channel A —protocol-related variable: <i>vRF!Header!NFIndicator</i> channel A. 0 <i>vRF!Header!NFIndicator</i> = 0 1 <i>vRF!Header!NFIndicator</i> = 1
SUA	Startup Frame Indicator Channel A —protocol-related variable: <i>vRF!Header!SuFIndicator</i> channel A. 0 <i>vRF!Header!SuFIndicator</i> = 0 1 <i>vRF!Header!SuFIndicator</i> = 1
SEA	Syntax Error on Channel A —protocol-related variable: <i>vSS!SyntaxError</i> channel A. 0 <i>vSS!SyntaxError</i> = 0 1 <i>vSS!SyntaxError</i> = 1
CEA	Content Error on Channel A —protocol-related variable: <i>vSS!ContentError</i> channel A. 0 <i>vSS!ContentError</i> = 0 1 <i>vSS!ContentError</i> = 1
BVA	Boundary Violation on Channel A —protocol-related variable: <i>vSS!BViolation</i> channel A. 0 <i>vSS!BViolation</i> = 0 1 <i>vSS!BViolation</i> = 1
TCA	Transmission Conflict on Channel A —protocol-related variable: <i>vSS!TxConflict</i> channel A. 0 <i>vSS!TxConflict</i> = 0 1 <i>vSS!TxConflict</i> = 1

51.5.2.49 Slot Status Counter Registers (FR_SSCR0–FR_SSCR3)

Figure 987. Slot Status Counter Registers (FR_SSCR0–FR_SSCR3)



Each of these four registers provides the slot status counter value for the previous communication cycle(s) and is updated at the cycle start. The provided value depends on the control bits and fields in the related internal Slot Status Counter Condition Register (FR_SSCCRn), which can be programmed by using the [Slot Status Counter Condition Register \(FR_SSCCR\)](#). For more details, see [Section 51.6.18.4, “Slot Status Counter Registers”](#).

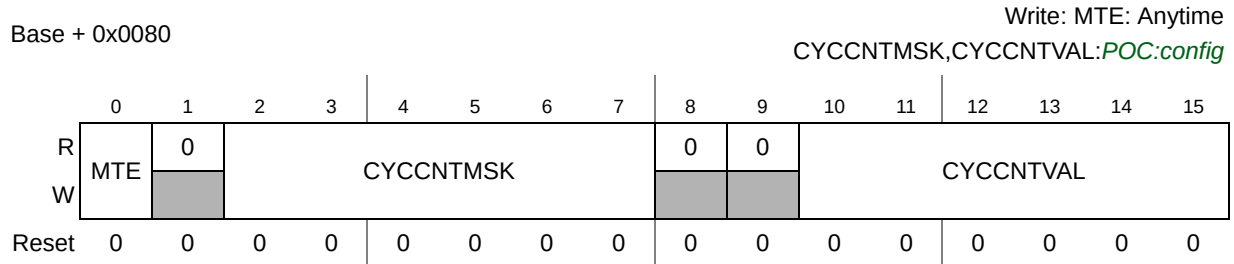
Note: If the counter has reached its maximum value 0xFFFF and is in Multicycle mode (that is, FR_SSCCRn[MCY] = 1), the counter is not reset to 0x0000. The application can reset the counter by clearing the FR_SSCCRn[MCY] bit and waiting for the next cycle start, when the CC clears the counter. Subsequently, the counter can be put into Multicycle mode again.

Table 1025. FR_SSCR0–FR_SSCR3 field descriptions

Field	Description
SLOTSTATUSCNT	Slot Status Counter —This field provides the current value of the Slot Status Counter.

51.5.2.50 MTS A Configuration Register (FR_MTSACFR)

Figure 988. MTS A Configuration Register (FR_MTSACFR)



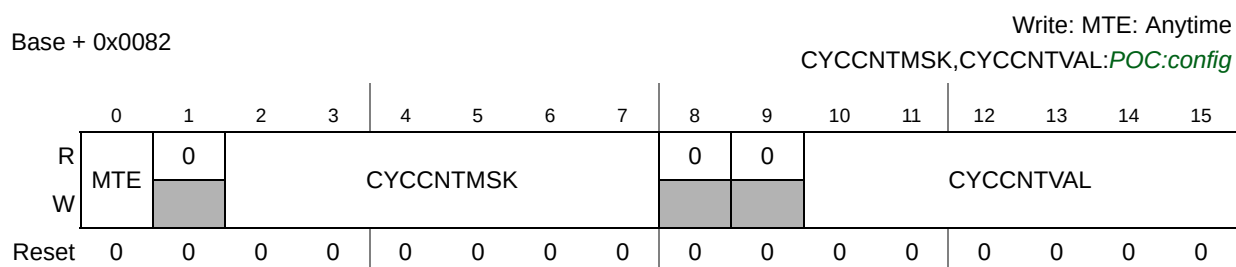
This register controls the transmission of the Media Access Test Symbol (MTS) on channel A. For more details, see [Section 51.6.13, “MTS generation”](#).

Table 1026. FR_MTSACFR field descriptions

Field	Description
MTE	Media Access Test Symbol Transmission Enable —This control bit is used to enable and disable the transmission of the MTS in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
CYCCNTMSK	Cycle Counter Mask—This field provides the filter mask for the MTS cycle count filter.
CYCCNTVAL	Cycle Counter Value —This field provides the filter value for the MTS cycle count filter.

51.5.2.51 MTS B Configuration Register (FR_MTSBCFR)

Figure 989. MTS B Configuration Register (FR_MTSBCFR)



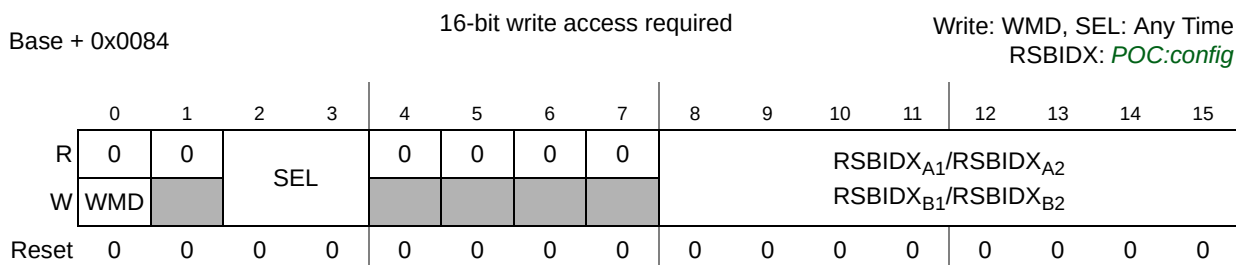
This register controls the transmission of the Media Access Test Symbol (MTS) on channel B. For more details, see [Section 51.6.13, “MTS generation”](#).

Table 1027. FR_MTSBCFR field descriptions

Field	Description
MTE	Media Access Test Symbol Transmission Enable —This control bit is used to enable and disable the transmission of the MTS in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
CYCCNTMSK	Cycle Counter Mask—This field provides the filter mask for the MTS cycle count filter.
CYCCNTVAL	Cycle Counter Value —This field provides the filter value for the MTS cycle count filter.

51.5.2.52 Receive Shadow Buffer Index Register (FR_RSBIR)

Figure 990. Receive Shadow Buffer Index Register (FR_RSBIR)



This register is used to provide and retrieve the indices of the message buffer header fields currently associated with the receive shadow buffers. For more details on the receive shadow buffer concept, refer to [Section 51.6.6.3.5, “Receive shadow buffer concept”](#).

Table 1028. FR_RSBIR field descriptions

Field	Description
WMD	Write Mode —This bit controls the write mode for this register. 0 Update SEL and RSBIDX field on register write 1 Update only SEL field on register write
SEL	Selector —This field is used to select the internal receive shadow buffer index register for access. 00 FR_RSBIR_A1—receive shadow buffer index register for channel A, segment 1 01 FR_RSBIR_A2—receive shadow buffer index register for channel A, segment 2 10 FR_RSBIR_B1—receive shadow buffer index register for channel B, segment 1 11 FR_RSBIR_B2—receive shadow buffer index register for channel B, segment 2
RSBIDX _{A1} RSBIDX _{A2} RSBIDX _{B1} RSBIDX _{B2}	Receive Shadow Buffer Index —This field contains the current index of the message buffer header field of the receive shadow message buffer selected by the SEL field. The CC uses this index to determine the physical location of the shadow buffer header field in the FlexRay memory area. The CC will update this field during receive operation. The application provides initial message buffer header index value in the configuration phase. CC: Updates the message buffer header index after successful reception. Application: Provides initial message buffer header index. Legal Values are $0 \leq i \leq 131$. Illegal values will be detected during the message buffer search.

51.5.2.53 Receive FIFO Start Data Offset Register (FR_RFSDOR)

Figure 991. Receive FIFO Start Data Offset Register (FR_RFSDOR)



Table 1029. FR_RFSDOR field descriptions

Field	Description
SDO _A SDO _B	Start Data Field Offset —This field defines the data field offset of the header field of the first message buffer of the selected FIFO. The CC uses the value of the SDO field to determine the physical location of the receiver FIFO's first message buffer header field. For configuration constraints, see Section 51.7.1.2, “Configure data field offsets” .

Note: Since all data fields of the FIFO are of equal length and are located at subsequent system memory addresses, the content of the FR_RFSDOR register corresponds to the start address of the payload area of the selected FIFO.

51.5.2.54 Receive FIFO System Memory Base Address Register (FR_RFSYMBADR)

Figure 992. Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)

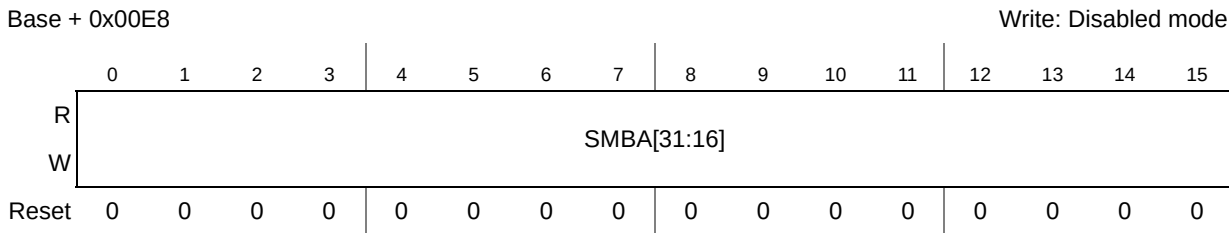
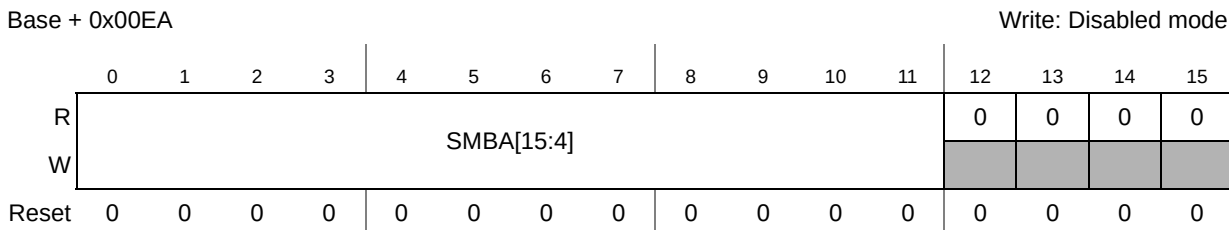


Figure 993. Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)



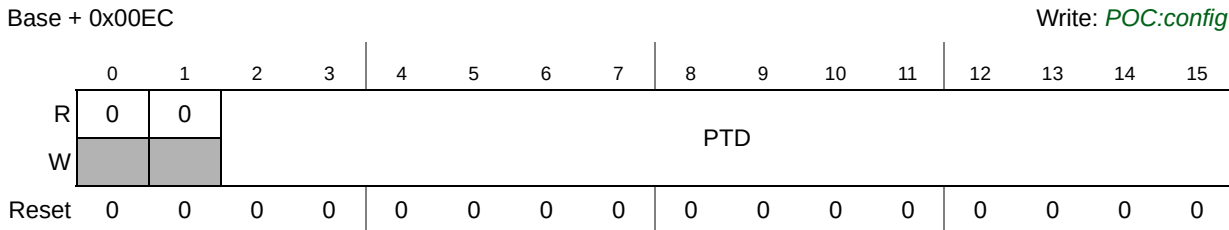
These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Table 1030. FR_RFSYMBADR field descriptions

Field	Description
SMBA	System Memory Base Address —This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defined as a byte address.

51.5.2.55 Receive FIFO Periodic Timer Register (FR_RFPTR)

Figure 994. Receive FIFO Periodic Timer Register (FR_RFPTR)



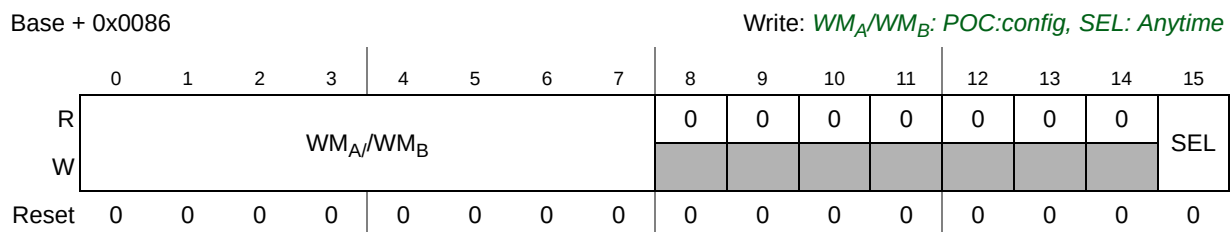
This register holds the periodic timer duration for the periodic FIFO timer. The periodic timer applies to both FIFOs (see [Section 51.6.9.3, "FIFO periodic timer"](#)).

Table 1031. FR_RFPTR field descriptions

Field	Description
PTD	Periodic Timer Duration —This value defines the periodic timer duration in terms of macroticks. 0000 Timer stays expired 3FFF Timer never expires other Timer expires after specified number of macroticks, expires and is restarted at each cycle start

51.5.2.56 Receive FIFO Watermark and Selection Register (FR_RFWMSR)

Figure 995. Receive FIFO Watermark and Selection Register (FR_RFWMSR)



This register is used to:

- Select a receiver FIFO for subsequent programming access through the receiver FIFO configuration registers summarized in [Table 1032](#).
- Define the watermark for the selected FIFO.

Table 1032. SEL-controlled receiver FIFO registers

Register
Receive FIFO Start Index Register (FR_RFSIR)
Receive FIFO Depth and Size Register (FR_RFDSR)
Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)
Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Table 1033. FR_RFWMSR field descriptions

Field	Description
WM _A WM _B	Watermark —This field defines the watermark value for the selected FIFO. This value is used to control the generation of the almost full interrupt flags.
SEL	Select —This control bit selects the receiver FIFO for subsequent programming. 0 Receiver FIFO for channel A selected 1 Receiver FIFO for channel B selected

51.5.2.57 Receive FIFO Start Index Register (FR_RFSIR)

Figure 996. Receive FIFO Start Index Register (FR_RFSIR)

Base + 0x0088

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	SIDX _A /SIDX _B									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines the message buffer header index of the first message buffer of the selected FIFO.

Table 1034. FR_RFSIR field descriptions

Field	Description
SIDX _A SIDX _B	Start Index —This field defines the number of the message buffer header field of the selected FIFO's first message buffer. The CC uses the value of the SIDX field to determine the physical location of the receiver FIFO's first message buffer header field.

51.5.2.58 Receive FIFO Depth and Size Register (FR_RFDSR)

Figure 997. Receive FIFO Depth and Size Register (FR_RFDSR)

Base + 0x008A

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIFO_DEPTH _A /FIFO_DEPTH _B								0	ENTRY_SIZE _A /ENTRY_SIZE _B						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

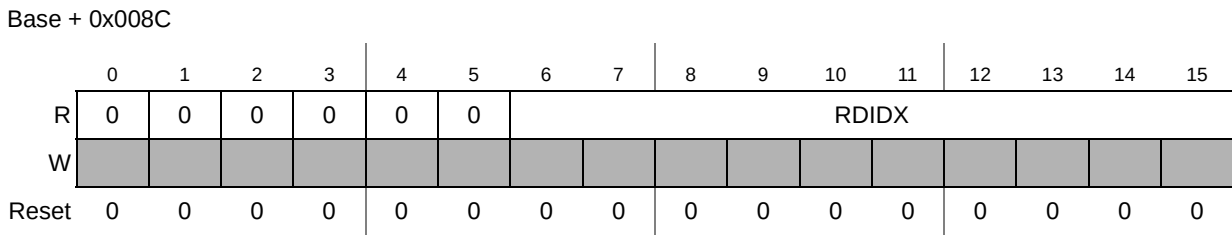
This register defines the structure of the selected FIFO, i.e. the number of entries and the size of each entry.

Table 1035. RFDSR Field Descriptions

Field	Description
FIFO_DEPTH _A FIFO_DEPTH _B	FIFO Depth —This field defines the depth (the number of entries) of the selected FIFO. Note: If the FIFO_DEPTH is configured to 0, FR_RFFIDRFMR[FIDRFMSK] must be configured to 0 as well to ensure that no frames are received into the FIFO.
ENTRY_SIZE _A ENTRY_SIZE _B	Entry Size —This field defines the size of the frame data sections for the selected FIFO in two-byte entities.

51.5.2.59 Receive FIFO A Read Index Register (FR_RFARIR)

Figure 998. Receive FIFO A Read Index Register (FR_RFARIR)



This register provides the message buffer header index of the next available FIFO A entry that the application can read.

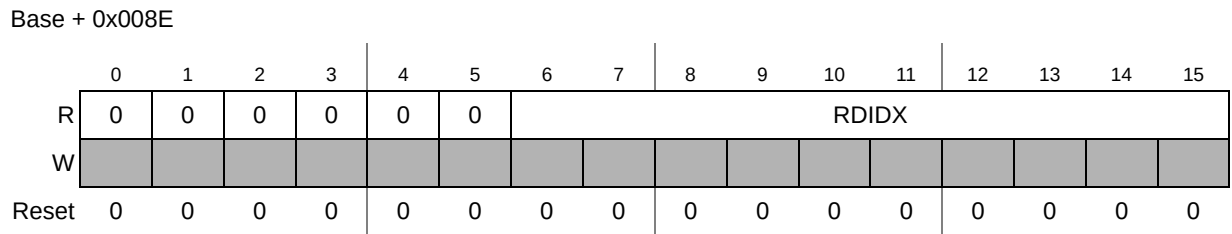
Table 1036. FR_RFARIR field descriptions

Field	Description
RDIDX	Read Index —This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

Note: If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

51.5.2.60 Receive FIFO B Read Index Register (FR_RFBRIR)

Figure 999. Receive FIFO B Read Index Register (FR_RFBRIR)



This register provides the message buffer header index of the next available FIFO B entry that the application can read.

Table 1037. FR_RFBRIR field descriptions

Field	Description
RDIDX	Read Index —This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

Note: If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

51.5.2.61 Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

Figure 1000. Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

Base + 0x00EE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FLB								FLA							
W	PCB								PCA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides the current fill level of the two receiver FIFOs and is used to pop a number of entries from the FIFOs.

Table 1038. FR_RFFLPCR field descriptions

Field	Description
FLB	Fill Level FIFO B —This field provides the current number of entries in the FIFO B.
FLA	Fill Level FIFO A —This field provides the current number of entries in the FIFO A.
PCB	Pop Count FIFO B —This field defines the number of entries to be removed from FIFO B.
PCA	Pop Count FIFO A —This field defines the number of entries to be removed from FIFO A.

Note: If the pop count value PCA/PCB is greater than the current FIFO fill level FLB/FLA, then the FIFO is empty after the update. No notification is given that not the required number of entries was removed.

51.5.2.62 Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

Figure 1001. Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

Base + 0x0090

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIDAFVAL _A /MIDAFVAL _B															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

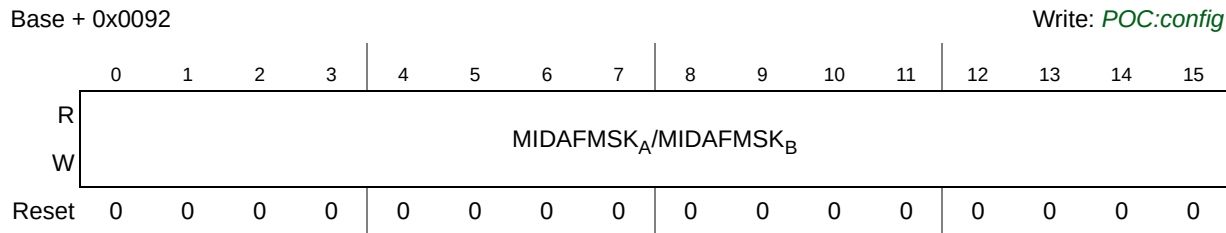
This register defines the filter value for the message ID acceptance filter of the selected FIFO. For details on message ID filtering, see [Section 51.6.9.9, “FIFO filtering”](#).

Table 1039. FR_RFMIDAFVR field descriptions

Field	Description
MIDAFVAL _A MIDAFVAL _B	Message ID Acceptance Filter Value —Filter value for the message ID acceptance filter.

51.5.2.63 Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

Figure 1002. Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)



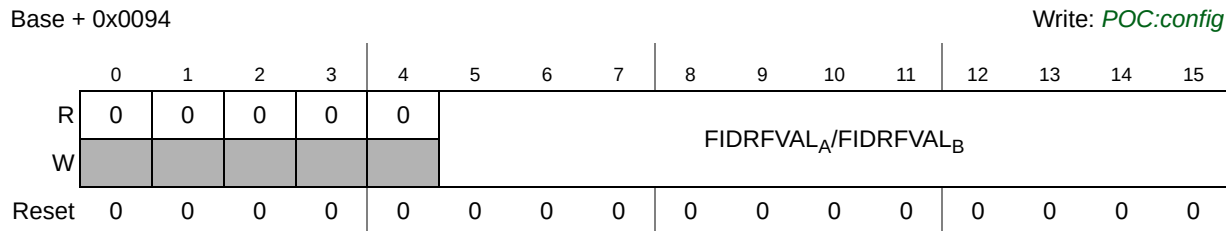
This register defines the filter mask for the message ID acceptance filter of the selected FIFO. For details on message ID filtering, see [Section 51.6.9.9, "FIFO filtering"](#).

Table 1040. FR_RFMIDAFMR field descriptions

Field	Description
MIDAFMSK _A MIDAFMSK _B	Message ID Acceptance Filter Mask —Filter mask for the message ID acceptance filter.

51.5.2.64 Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

Figure 1003. Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)



This register defines the filter value for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering, see [Section 51.6.9.9, "FIFO filtering"](#).

Table 1041. FR_RFFIDRFVR field descriptions

Field	Description
FIDRFVAL _A FIDRFVAL _B	Frame ID Rejection Filter Value —Filter value for the frame ID rejection filter.

51.5.2.65 Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

Figure 1004. Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

Base + 0x0096

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	FIDRFMSK _A /FIDRFMSK _B										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines the filter mask for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see [Section 51.6.9.9, "FIFO filtering"](#).

Table 1042. FR_RFFIDRFMR field descriptions

Field	Description
FIDRFMSK	Frame ID Rejection Filter Mask—Filter mask for the frame ID rejection filter.

51.5.2.66 Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Figure 1005. Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Base + 0x0098

16-bit write access required

Write: WMD, IBD, SEL: Any Time
SID: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				0	SID _A /SID _B										
W	WMD	IBD	SEL													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides access to the four internal frame ID range filter boundary registers of the selected FIFO. For details on frame ID range filter, see [Section 51.6.9.9, "FIFO filtering"](#).

Table 1043. FR_RFRFCFR field descriptions

Field	Description
WMD	Write Mode —This control bit defines the write mode of this register. 0 Write to all fields in this register on write access 1 Write to SEL and IBD field only on write access
IBD	Interval Boundary —This control bit selects the interval boundary to be programmed with the SID value. 0 Program lower interval boundary 1 Program upper interval boundary

Table 1043. FR_RFRFCFR field descriptions

Field	Description
SEL	Filter Selector —This control field selects the frame ID range filter to be accessed. 00 Select frame ID range filter 0 01 Select frame ID range filter 1 10 Select frame ID range filter 2 11 Select frame ID range filter 3
SID _A SID _B	Slot ID —Defines the IBD-selected frame ID boundary value for the SEL-selected range filter.

51.5.2.67 Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Figure 1006. Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Base + 0x009A

Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	F3M	F2M	F1M	F0M	0	0	0	0	F3EN	F2EN	F1EN	F0EN
W					D	D	D	D								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register is used to enable and disable each frame ID range filter and to define whether it is running as an acceptance or rejection filter.

Table 1044. FR_RFRFCTR field descriptions

Field	Description
F3MD	Range Filter 3 Mode —This control bit defines the filter mode of the frame ID range filter 3. 0 Range filter 3 runs as acceptance filter 1 Range filter 3 runs as rejection filter
F2MD	Range Filter 2 Mode —This control bit defines the filter mode of the frame ID range filter 2. 0 Range filter 2 runs as acceptance filter 1 Range filter 2 runs as rejection filter
F1MD	Range Filter 1 Mode —This control bit defines the filter mode of the frame ID range filter 1. 0 Range filter 1 runs as acceptance filter 1 Range filter 1 runs as rejection filter
F0MD	Range Filter 0 Mode —This control bit defines the filter mode of the frame ID range filter 0. 0 Range filter 0 runs as acceptance filter 1 Range filter 0 runs as rejection filter
F3EN	Range Filter 3 Enable —This control bit is used to enable and disable the frame ID range filter 3. 0 Range filter 3 disabled 1 Range filter 3 enabled
F2EN	Range Filter 2 Enable —This control bit is used to enable and disable the frame ID range filter 2. 0 Range filter 2 disabled 1 Range filter 2 enabled

Table 1044. FR_RFRFCTR field descriptions (continued)

Field	Description
F1EN	Range Filter 1 Enable —This control bit is used to enable and disable the frame ID range filter 1. 0 Range filter 1 disabled 1 Range filter 1 enabled
F0EN	Range Filter 0 Enable —This control bit is used to enable and disable the frame ID range filter 0. 0 Range filter 0 disabled 1 Range filter 0 enabled

51.5.2.68 Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

Figure 1007. Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

Base + 0x009C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	LASTDYNTXSLOTA										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides the number of the last transmission slot in the dynamic segment for channel A. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Table 1045. FR_LDTXSLAR field descriptions

Field	Description
LASTDYNTXSLOTA	Last Dynamic Transmission Slot Channel A —protocol-related variable: zLastDynTxSlot channel A. Number of the last transmission slot in the dynamic segment for channel A. If no frame was transmitted during the dynamic segment on channel A, the value of this field is set to 0.

51.5.2.69 Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

Figure 1008. Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

Base + 0x009E

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	LASTDYNTXSLOTB										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides the number of the last transmission slot in the dynamic segment for channel B. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Table 1046. FR_LDTXSLBR field descriptions

Field	Description
LASTDYNTX SLOTB	Last Dynamic Transmission Slot Channel B —protocol-related variable: zLastDynTxSlot channel B. Number of the last transmission slot in the dynamic segment for channel B. If no frame was transmitted during the dynamic segment on channel B the value of this field is set to 0.

51.5.2.70 Protocol Configuration Registers

The following configuration registers provide the necessary configuration information to the protocol engine. The individual values in the registers are described in [Table 1047](#). For more details about the FlexRay related configuration parameters and the allowed parameter ranges, see *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

Table 1047. Protocol Configuration Register fields

Name	Description ⁽¹⁾	Min	Max	Unit	FR_PCR
coldstart_attempts	gColdstartAttempts			number	3
action_point_offset	gdActionPointOffset - 1			MT	0
cas_rx_low_max	gdCASRxLowMax - 1			gdBit	4
dynamic_slot_idle_phase	gdDynamicSlotIdlePhase			minislot	28
minislot_action_point_offset	gdMinislotActionPointOffset - 1			MT	3
minislot_after_action_point	gdMinislot - gdMinislotActionPointOffset - 1			MT	2
static_slot_length	gdStaticSlot			MT	0
static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1			MT	13
symbol_window_exists	gdSymbolWindow !=0	0	1	bool	9
symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1			MT	6
tss_transmitter	gdTSSTransmitter			gdBit	5
wakeup_symbol_rx_idle	gdWakeupSymbolRxIdle			gdBit	5
wakeup_symbol_rx_low	gdWakeupSymbolRxLow			gdBit	3
wakeup_symbol_rx_window	gdWakeupSymbolRxWindow			gdBit	4
wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle			gdBit	8
wakeup_symbol_tx_low	gdWakeupSymbolTxLow			gdBit	5
noise_listen_timeout	(gListenNoise * pdListenTimeout) - 1			μT	16/17
macro_initial_offset_a	pMacroInitialOffset[A]			MT	6
macro_initial_offset_b	pMacroInitialOffset[B]			MT	16
macro_per_cycle	gMacroPerCycle			MT	10
macro_after_first_static_slot	gMacroPerCycle - gdStaticSlot			MT	1
macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart			MT	28

Table 1047. Protocol Configuration Register fields (continued)

Name	Description ⁽¹⁾	Min	Max	Unit	FR_PCR
max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal			cyclepairs	8
max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive			cyclepairs	8
minislot_exists	gNumberOfMinislots != 0	0	1	bool	9
minislots_max	gNumberOfMinislots - 1			minislot	29
number_of_static_slots	gNumberOfStaticSlots			static slot	2
offset_correction_start	gOffsetCorrectionStart			MT	11
payload_length_static	gPayloadLengthStatic			2 bytes	19
max_payload_length_dynamic	pPayloadLengthDynMax			2 bytes	24
first_minislot_action_point_offset	max(gdActionPointOffset , gdMinislotActionPointOffset) - 1			MT	13
allow_halt_due_to_clock	pAllowHaltDueToClock			bool	26
allow_passive_to_active	pAllowPassiveToActive			cyclepairs	12
cluster_drift_damping	pClusterDriftDamping			μT	24
comp_accepted_startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]			μT	22
comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]			μT	26
listen_timeout	pdListenTimeout - 1			μT	14/15
key_slot_id	pKeySlotId			number	18
key_slot_used_for_startup	pKeySlotUsedForStartup			bool	11
key_slot_used_for_sync	pKeySlotUsedForSync			bool	11
latest_tx	gNumberOfMinislots - pLatestTx			minislot	21
sync_node_max	gSyncNodeMax			number	30
micro_initial_offset_a	pMicroInitialOffset[A]			μT	20
micro_initial_offset_b	pMicroInitialOffset[B]			μT	20
micro_per_cycle	pMicroPerCycle			μT	22/23
micro_per_cycle_min	pMicroPerCycle - pdMaxDrift			μT	24/25
micro_per_cycle_max	pMicroPerCycle + pdMaxDrift			μT	26/27
micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)			μT	7
offset_correction_out	pOffsetCorrectionOut			μT	9
rate_correction_out	pRateCorrectionOut			μT	14
single_slot_enabled	pSingleSlotEnabled			bool	10
wakeup_channel	pWakeupChannel	see Table 1048			10
wakeup_pattern	pWakeupPattern			number	18
decoding_correction_a	pDecodingCorrection + pDelayCompensation[A] + 2			μT	19

Table 1047. Protocol Configuration Register fields (continued)

Name	Description ⁽¹⁾	Min	Max	Unit	FR_PCR
decoding_correction_b	$pDecodingCorrection + pDelayCompensation[B] + 2$			μT	7
key_slot_header_crc	header CRC for key slot	0x000	0x7FF	number	12
extern_offset_correction	pExternOffsetCorrection			μT	29
extern_rate_correction	pExternRateCorrection			μT	21

1. See *FlexRay Communications System Protocol Specification, Version 2.1 Rev A* for detailed protocol parameter definitions

Table 1048. Wakeup channel selection

wakeup_channel	Wakeup Channel
0	A
1	B

51.5.2.70.1 Protocol Configuration Register 0 (FR_PCR0)

Figure 1009. Protocol Configuration Register 0 (FR_PCR0)

Base + 0x00A0

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	action_point_offset				static_slot_length											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.2 Protocol Configuration Register 1 (FR_PCR1)

Figure 1010. Protocol Configuration Register 1 (FR_PCR1)

Base + 0x00A2

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	macro_after_first_static_slot													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.3 Protocol Configuration Register 2 (FR_PCR2)

Figure 1011. Protocol Configuration Register 2 (FR_PCR2)

Base + 0x00A4

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	minislot_after_action_point								number_of_static_slots							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.4 Protocol Configuration Register 3 (FR_PCR3)

Figure 1012. Protocol Configuration Register 3 (FR_PCR3)

Base + 0x00A6

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	wakeup_symbol_rx_low								minislot_action_point_offset[4:0]				coldstart_attempts			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.5 Protocol Configuration Register 4 (FR_PCR4)

Figure 1013. Protocol Configuration Register 4 (FR_PCR4)

Base + 0x00A8

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	cas_rx_low_max								wakeup_symbol_rx_window							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.6 Protocol Configuration Register 5 (FR_PCR5)

Figure 1014. Protocol Configuration Register 5 (FR_PCR5)

Base + 0x00AA

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	tss_transmitter				wakeup_symbol_tx_low								wakeup_symbol_rx_idle			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.7 Protocol Configuration Register 6 (FR_PCR6)

Figure 1015. Protocol Configuration Register 6 (FR_PCR6)

Base + 0x00AC

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	symbol_window_after_action_point							macro_initial_offset_a							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.8 Protocol Configuration Register 7 (FR_PCR7)

Figure 1016. Protocol Configuration Register 7 (FR_PCR7)

Base + 0x00AE

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	decoding_correction_b							micro_per_macro_nom_half								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.9 Protocol Configuration Register 8 (FR_PCR8)

Figure 1017. Protocol Configuration Register 8 (FR_PCR8)

Base + 0x00B0

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	max_without_clock_ correction_fatal			max_without_clock_ correction_passive				wakeup_symbol_tx_idle								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.10 Protocol Configuration Register 9 (FR_PCR9)

Figure 1018. Protocol Configuration Register 9 (FR_PCR9)

Base + 0x00B2

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	mini slot_ exists	sym bol_ win dow_ exists	offset_correction_out													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.11 Protocol Configuration Register 10 (FR_PCR10)

Figure 1019. Protocol Configuration Register 10 (FR_PCR10)

Base + 0x00B4

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	single_slot_en_abled	wake_up_channel	macro_per_cycle													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.12 Protocol Configuration Register 11 (FR_PCR11)

Figure 1020. Protocol Configuration Register 11 (FR_PCR11)

Base + 0x00B6

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	key_slot_used_for_start_up	key_slot_used_for_sync	offset_correction_start													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.13 Protocol Configuration Register 12 (FR_PCR12)

Figure 1021. Protocol Configuration Register 12 (FR_PCR12)

Base + 0x00B8

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	allow_passive_to_active				key_slot_header_crc											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.14 Protocol Configuration Register 13 (FR_PCR13)

Figure 1022. Protocol Configuration Register 13 (FR_PCR13)

Base + 0x00BA

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	first_minislot_action_point_offset				static_slot_after_action_point											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.15 Protocol Configuration Register 14 (FR_PCR14)

Figure 1023. Protocol Configuration Register 14 (FR_PCR14)

Base + 0x00BC

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	rate_correction_out												listen_timeout[20:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.16 Protocol Configuration Register 15 (FR_PCR15)

Figure 1024. Protocol Configuration Register 15 (FR_PCR15)

Base + 0x00BE

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	listen_timeout[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.17 Protocol Configuration Register 16 (FR_PCR16)

Figure 1025. Protocol Configuration Register 16 (FR_PCR16)

Base + 0x00C0

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	macro_initial_offset_b								noise_listen_timeout[24:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.18 Protocol Configuration Register 17 (FR_PCR17)

Figure 1026. Protocol Configuration Register 17 (FR_PCR17)

Base + 0x00C2

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	noise_listen_timeout[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.19 Protocol Configuration Register 18 (FR_PCR18)

Figure 1027. Protocol Configuration Register 18 (FR_PCR18)

Base + 0x00C4

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	wakeup_pattern								key_slot_id							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.20 Protocol Configuration Register 19 (FR_PCR19)

Figure 1028. Protocol Configuration Register 19 (FR_PCR19)

Base + 0x00C6

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	decoding_correction_a								payload_length_static							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.21 Protocol Configuration Register 20 (FR_PCR20)

Figure 1029. Protocol Configuration Register 20 (FR_PCR20)

Base + 0x00C8

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	micro_initial_offset_b								micro_initial_offset_a							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.22 Protocol Configuration Register 21 (FR_PCR21)

Figure 1030. Protocol Configuration Register 21 (FR_PCR21)

Base + 0x00CA

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	extern_rate_correction				latest_tx											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.23 Protocol Configuration Register 22 (FR_PCR22)

Figure 1031. Protocol Configuration Register 22 (FR_PCR22)

Base + 0x00CC

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R* comp_accepted_startup_range_a micro_per_cycle[19:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.24 Protocol Configuration Register 23 (FR_PCR23)

Figure 1032. Protocol Configuration Register 23 (FR_PCR23)

Base + 0x00CE

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	micro_per_cycle[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.25 Protocol Configuration Register 24 (FR_PCR24)

Figure 1033. Protocol Configuration Register 24 (FR_PCR24)

Base + 0x00D0

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	cluster_drift_damping				max_payload_length_dynamic								micro_per_cycle_min [19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.26 Protocol Configuration Register 25 (FR_PCR25)

Figure 1034. Protocol Configuration Register 25 (FR_PCR25)

Base + 0x00D2

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	micro_per_cycle_min[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.27 Protocol Configuration Register 26 (FR_PCR26)

Figure 1035. Protocol Configuration Register 26 (FR_PCR26)

Base + 0x00D4

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	allow_halt_due_to_clock	comp_accepted_startup_range_b											micro_per_cycle_max [19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.28 Protocol Configuration Register 27 (FR_PCR27)

Figure 1036. Protocol Configuration Register 27 (FR_PCR27)

Base + 0x00D6

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	micro_per_cycle_max[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.29 Protocol Configuration Register 28 (FR_PCR28)

Figure 1037. Protocol Configuration Register 28 (FR_PCR28)

Base + 0x00D8

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	dynamic_slot_idle_phase		macro_after_offset_correction													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.30 Protocol Configuration Register 29 (FR_PCR29)

Figure 1038. Protocol Configuration Register 29 (FR_PCR29)

Base + 0x00DA

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	extern_offset_correction				minislots_max											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.70.31 Protocol Configuration Register 30 (FR_PCR30)

Figure 1039. Protocol Configuration Register 30 (FR_PCR30)

Base + 0x00DC

Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	sync_node_max			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51.5.2.71 ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

Figure 1040. ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

Base + 0x00F0

In Functional Test Mode writing 1 toggles the _OF and _IF flags.

Write: Normal Mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LRNE_OF	LRCE_OF	DRNE_OF	DRCE_OF	LRNE_IF	LRCE_IF	DRNE_IF	DRCE_IF	0	0	0	0	LRNE_IE	LRCE_IE	DRNE_IE	DRCE_IE
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides the means to control the ECC-related interrupt request lines and provides the corresponding interrupt flags. The interrupt flags are cleared by writing 1, which resets the corresponding report registers. For a detailed description, see [Section 51.6.24.2, “Memory error reporting.”](#)

Table 1049. FR_EEIFER field descriptions

Field	Description
Error Overflow Flags	
LRNE_OF	LRAM Non-Corrected Error Overflow Flag—This flag is set to 1 when at least one of the following events occurs: a) Memory errors are detected <i>but not corrected</i> on CHI LRAM and interrupt flag LRNE_IF is already 1 b) Memory errors are detected <i>but not corrected</i> on at least two banks of CHI LRAM 0 No such event 1 Non-Corrected Error overflow detected on CHI LRAM
LRCE_OF	LRAM Corrected Error Overflow Flag—This flag is set to 1 when at least one of the following events occurs: a) Memory errors are detected <i>and corrected</i> on CHI LRAM and interrupt flag LRCE_IF is already 1 b) Memory errors are detected <i>and corrected</i> on at least two banks of CHI LRAM 0 No such event 1 Corrected Error overflow detected on CHI LRAM Note: Error Correction is not implemented on CHI LRAM, so the flag will never be asserted.
DRNE_OF	DRAM Non-Corrected Error Overflow Flag—This flag is set to 1 when at least one of the following events appears: a) Memory errors are detected <i>but not corrected</i> on PE DRAM and interrupt flag DRNE_IF is already 1 b) Memory errors are detected <i>but not corrected</i> on at least two banks of the PE DRAM 0 No such event 1 Non-Corrected Error overflow detected on PE DRAM
DRCE_OF	DRAM Corrected Error Overflow Flag—This flag is set to 1 when at least one of the following events appears: a) Memory errors are detected <i>and corrected</i> on PE DRAM and interrupt flag DRCE_IF is already 1 b) Memory errors are detected <i>and corrected</i> on at least two banks of PE DRAM 0 No such event 1 Corrected Error overflow detected on PE DRAM

Table 1049. FR_EEIFER field descriptions (continued)

Field	Description
Error Interrupt Flags	
LRNE_IF	LRAM Non-Corrected Error Interrupt Flag —This interrupt flag is set to 1 when a memory error is <i>detected but not corrected</i> on the CHI LRAM. 0 No such event 1 Non-Corrected Error detected on CHI LRAM
LRCE_IF	LRAM Corrected Error Interrupt Flag —This interrupt flag is set to 1 when a memory error is <i>detected and corrected</i> on the CHI LRAM. 0 No such event 1 Corrected Error detected on CHI LRAM Note: Error Correction is not implemented on CHI LRAM, so the flag will never be asserted.
DRNE_IF	DRAM Non-Corrected Error Interrupt Flag —This interrupt flag is set to 1 when a memory error is <i>detected but not corrected</i> on PE DRAM. 0 No such event 1 Non-Corrected Error detected on PE DRAM
DRCE_IF	DRAM Corrected Error Interrupt Flag —This interrupt flag is set to 1 when a memory error is <i>detected and corrected</i> on PE DRAM. 0 No such event 1 Corrected Error detected on PE DRAM
Error Interrupt Enables	
LRNE_IE	LRAM Non-Corrected Error Interrupt Enable —This flag controls if the LRAM Non-Corrected Error Interrupt line is asserted when the LRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
LRCE_IE	LRAM Corrected Error Interrupt Enable —This flag controls if the LRAM Corrected Error Interrupt line is asserted when the LRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
DRNE_IE	DRAM Non-Corrected Error Interrupt Enable —This flag controls if the DRAM Non-Corrected Error Interrupt line is asserted when the DRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
DRCE_IE	DRAM Corrected Error Interrupt Enable —This flag controls if the DRAM Corrected Error Interrupt line is asserted when the DRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line

51.5.2.72 ECC Error Report and Injection Control Register (FR_EERICR)

Figure 1041. ECC Error Report and Injection Control Register (FR_EERICR)

Base + 0x00F2

Write: ERS: Anytime

ERM, EIM, EIE: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	0	0	0	0	0	ERS		0	0	0	ERM	0	0	EIM	EIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register configures the error injection and error reporting and provides the selector for the content of the report registers.

Table 1050. FR_EERICR Field Descriptions

Field	Description
BSY	Register Update Busy —This field indicates the current state of the ECC configuration update and controls the register write access condition IDL specified in “ Section 51.5.2.2, “Register write access.” 0 ECC configuration is idle 1 ECC configuration is running
ERS	Error Report Select —This field selects the content of the ECC Error reporting registers. 00 Show PE DRAM non-corrected error information 01 Show PE DRAM corrected error information 10 Show CHI LRAM non-corrected error information 11 Show CHI LRAM corrected error information
ERM	Error Report Mode —This bit configures the type of data written into the internal error report registers on the detection of a memory error. 0 Store data and code as delivered by ecc decoding logic 1 Store data and code as read from the memory
EIM	Error Injection Mode —This bit configures the ECC error injection mode. 0 Use FR_EEIDR[DATA] and FR_EEICR[CODE] as XOR distortion pattern for error injection 1 Use FR_EEIDR[DATA] and FR_EEICR[CODE] as write value for error injection
EIE	Error Injection Enable —This bit configures the ECC error injection on the memories. 0 Error injection disabled 1 Error injection enabled

51.5.2.73 ECC Error Report Address Register (FR_EERAR)

Figure 1042. ECC Error Report Address Register (FR_EERAR)

Base + 0x00F4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MID	BANK			ADDR											
W																
Reset	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

This register provides the memory identifier, bank, and address for which the memory error is reported.

Table 1051. FR_EERAR Field Descriptions

Field	Description
MID	Memory Identifier —This flag provides the memory instance for which the memory error is reported. 0 PE DRAM 1 CHI LRAM
BANK	Memory Bank —This field provides the BANK for which the memory error is reported. 111 reset value, indicates no error found after reset. For MID = 0: 000 PE DRAM [7:0] 001 PE DRAM [15:8] Others—not used For MID = 1: Refer to Table 1052 for the assignment of the LRAM banks.
ADDR	Memory Address —This field provides the address of the failing memory location.

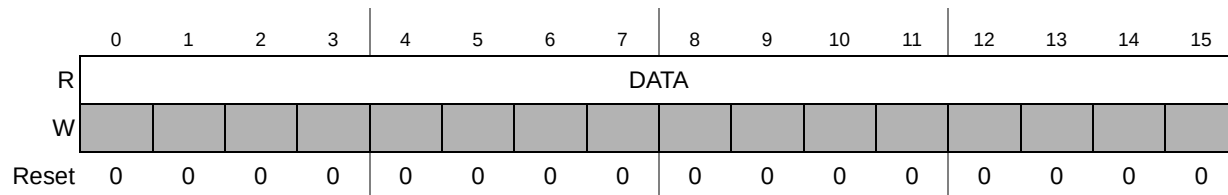
Table 1052. LRAM bank value for MID = 1

BANK	Register		
000	FR_MBCCFR(2n)	FR_MBDOR(6n)	FR_LEETR0
001	FR_MBFIDR(2n)	FR_MBDOR(6n + 1)	FR_LEETR1
010	FR_MBIDXR(2n)	FR_MBDOR(6n + 2)	FR_LEETR2
011	FR_MBCCFR(2n+1)	FR_MBDOR(6n + 3)	FR_LEETR3
100	FR_MBFIDR(2n+1)	FR_MBDOR(6n + 4)	FR_LEETR4
101	FR_MBIDXR(2n+1)	FR_MBDOR(6n + 5)	FR_LEETR5
110	Not Used	Not Used	Not Used
111			

51.5.2.74 ECC Error Report Data Register (FR_EERDR)

Figure 1043. ECC Error Report Data Register (FR_EERDR)

Base + 0x00F6



This register provides the data related information of the reported memory read access. The assignment of the bits depends on the selected memory and memory bank as shown in [Table 1054](#).

Table 1053. FR_EERDR field descriptions

Field	Description
DATA	Data —The content of this field depends on the report mode selected by FR_EERICR[ERM]. ERM = 0: Ecc Data, shows data as generated by the ecc decoding logic ERM = 1: Memory Data, shows data as read from the memory

Table 1054. Valid bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE DRAM	0									PE DRAM[7:0]							
PE DRAM	1									PE DRAM[15:8]							
CHI LRAM	0	FR_MBCCFR(2n)															
CHI LRAM	0	FR_MBDOR(6n)															
CHI LRAM	0	FR_LEETR0															
CHI LRAM	1						FR_MBFIDR(2n)[FID]										
CHI LRAM	1	FR_MBDOR(6n+1)															
CHI LRAM	1	FR_LEETR1															
CHI LRAM	2									FR_MBIDXR(2n)[MBIDX]							
CHI LRAM	2	FR_MBDOR(6n+2)															
CHI LRAM	2	FR_LEETR2															
CHI LRAM	3	FR_MBCCFR(2n+1)															
CHI LRAM	3	FR_MBDOR(6n+3)															
CHI LRAM	3	FR_LEETR3															
CHI LRAM	4						FR_MBFIDR(2n+1)[FID]										
CHI LRAM	4	FR_MBDOR(6n+4)															
CHI LRAM	4	FR_LEETR4															
CHI LRAM	5									FR_MBIDXR(2n+1)[MBIDX]							

Table 1054. Valid bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHI LRAM	5	FR_MBDOR(6n+5)															
CHI LRAM	5	FR_LEETR5															

51.5.2.75 ECC Error Report Code Register (FR_EERCR)

Figure 1044. ECC Error Report Code Register (FR_EERCR)

Base + 0x00F8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	CODE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register provides the ECC-related information of the reported memory read access.

Table 1055. FR_EERSR field descriptions

Field	Description
CODE	Code —The content of this field depends on the report mode selected by FR_EERICR[ERM]. ERM = 0: Syndrome. Shows the ECC syndrome generated by the ECC decoding logic. The coding of the PE DRAM syndrome is shown in Section 51.6.24.2.2: PE DRAM syndrome . The coding of the CHI LRAM syndrome is shown in Section 51.6.24.2.4: CHI LRAM syndrome . ERM = 1: Checkbits. Shows the ecc checkbits read from the memory.

51.5.2.76 ECC Error Injection Address Register (FR_EEIAR)

Figure 1045. ECC Error Injection Address Register (FR_EEIAR)

Base + 0x00FA

Write: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	MID	BANK			ADDR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

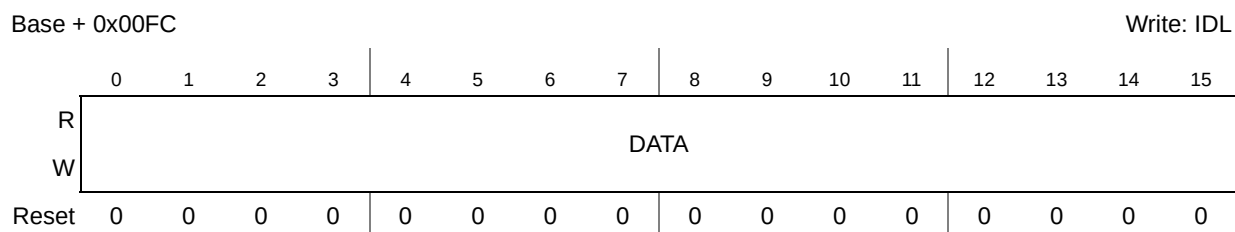
This register defines the memory module, bank, and address where the ECC error has to be injected.

Table 1056. FR_EEIAR field descriptions

Field	Description
MID	Memory Identifier —This flag defines the memory instance for ECC error injection. 0 PE DRAM 1 CHI LRAM
BANK	Memory Bank —This field defines the memory bank for ECC error injection. For MID = 0: 000 BANK0: PE DRAM [7:0] 001 BANK1: PE DRAM [15:8] Others reserved For MID = 1: Refer to Table 1052 for the assignment of the LRAM banks.
ADDR	Memory Address —This flag defines the memory address for ECC error injection.

51.5.2.77 ECC Error Injection Data Register (FR_EEIDR)

Figure 1046. ECC Error Injection Data Register (FR_EEIDR)



This register defines the data distortion pattern for the error injection write. The number of valid bits depends on the selected memory and memory bank as shown in [Table 1054](#).

Table 1057. FR_EEIDR field descriptions

Field	Description
DATA	Data —The content of this field depends on the error injection mode selected by FR_EEICR[EIM]. EIM = 0: This field defines the XOR distortion pattern for the data written into the memory. EIM = 1: This field defines the data to be written into the memory.

Note: The effect of the error injected depends from the LRAM content at the address accessed and from the module internal usage of the data. Refer to [Section 51.6.24.3, "Memory error response"](#) for details.

51.5.2.78 ECC Error Injection Code Register (FR_EEICR)

Figure 1047. ECC Error Injection Code Register (FR_EEICR)

Base + 0x00FE												Write: IDL				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	CODE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines the ECC code distortion pattern for the error injection write.

Table 1058. FR_EEICR field descriptions

Field	Description
CODE	Code —The content of this field depends on the error injection mode selected by FR_EEICR[EIM]. EIM = 0: This field defines the XOR distortion pattern for the ECC checkbits written into the memory. EIM = 1: This field defines the ECC checkbits written into the memory.

51.5.2.79 Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)

Figure 1048. Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)

Base + 0x0800 (FR_MBCCSR0)	Write: MTD: <i>POC:cnfig</i> or MB_DIS CMT: MB_LCK or MB_DIS EDT, LCKT, MBIE, MBIF: Normal mode
Base + 0x0808 (FR_MBCCSR1)	
...	
Base + 0x0BF8 (FR_MBCCSR127)	

Additional Reset: CMT, DUP, DVAL, MBIF: Message Buffer Disable

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	MTD	CMT	0	0	MBIE	0	0	0	DUP	DVAL	EDS	LCKS	MBIF
W					rwm	EDT	LCKT									w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The content of these registers comprises message buffer configuration data, message buffer control data, message buffer status information, and message buffer interrupt flags. A detailed description of all flags can be found in [Section 51.6.6, “Individual message buffer functional description”](#).

If the application writes 1 to the EDT bit, no write access to the other register bits is performed.

If the application writes 0 to the EDT bit and 1 to the LCKT bit, no write access to the other bits is performed.

Table 1059. FR_MBCCSRn field descriptions

Field	Description
Message Buffer Configuration	
MTD	Message Buffer Transfer Direction —This bit configures the transfer direction of a message buffer. 0 Receive message buffer 1 Transmit message buffer
Message Buffer Control	
CMT	Commit for Transmission —This bit indicates if the transmit message buffer data are ready for transmission. 0 Message buffer data not ready for transmission 1 Message buffer data ready for transmission
EDT	Enable/Disable Trigger —If the application writes 1 to this bit, a message buffer enable or disable is triggered, depending on the current value of the EDS status bit. 0 No effect 1 Message buffer enable or disable is triggered
LCKT	Lock/Unlock Trigger —If the application writes 1 to this bit and writes 0 to the EDT bit, a message buffer lock or unlock is triggered, depending on the current value of the LCKS status bit. 0 No effect 1 Message buffer lock or unlock is triggered
MBIE	Message Buffer Interrupt Enable —This control bit defines whether the message buffer will generate an interrupt request when its MBIF flag is set. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
Message Buffer Status	
DUP	Data Updated —This status bit indicates whether the frame header in the message buffer header field and the data in the message buffer data field were updated after a frame reception. 0 Frame Header and message buffer data field not updated 1 Frame Header and message buffer data field updated
DVAL	Data Valid —For receive message buffers this status bit indicates whether the message buffer data field contains valid frame data. For transmit message buffers, the status bit indicates if a message is transferred again due to the state transmission mode of the message buffer. 0 Receive message buffer contains no valid frame data / message is transmitted for the first time 1 Receive message buffer contains valid frame data / message will be transferred again
EDS	Enable/Disable Status —This status bit indicates whether the message buffer is enabled or disabled. 0 Message buffer is disabled 1 Message buffer is enabled
LCKS	Lock Status —This status bit indicates the current lock status of the message buffer. 0 Message buffer is not locked by the application 1 Message buffer is locked by the application
MBIF	Message Buffer Interrupt Flag —This flag is set when the slot status field of the message buffer was updated after frame transmission or reception, or when a transmit message buffer was just enabled by the application. 0 No such event 1 Slot status field updated or transmit message buffer just enabled In Functional Test Mode writing a 1 toggles the interrupt flag.

51.5.2.80 Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)

Figure 1049. Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)

Base + 0x0802 (FR_MBCCFR0)

Base + 0x080A (FR_MBCCFR1)

16-bit write access required

Write: *POC:config* or MB_DIS

...

Base + 0x0BFA (FR_MBCCFR127)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MTM	CHA	CHB	CCF	CCFMSK					CCFVAL						
W				E												
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This register contains message buffer configuration data for the transmission mode, the channel assignment, and for the cycle counter filtering. For detailed information on cycle counter filtering, refer to [Section 51.6.7.1, “Message buffer cycle counter filtering”](#).

Table 1060. FR_MBCCFRn field descriptions

Field	Description
MTM	Message Buffer Transmission Mode —This control bit applies only to transmit message buffers and defines the transmission mode. 0 Event Transmission mode 1 State Transmission mode
CHA CHB	Channel Assignment —These control bits define the channel assignment and control the receive and transmit behavior of the message buffer according to Table 1061 .
CCFE	Cycle Counter Filtering Enable —This control bit is used to enable and disable the cycle counter filtering. 0 Cycle counter filtering disabled 1 Cycle counter filtering enabled
CCFMSK	Cycle Counter Filtering Mask —This field defines the filter mask for the cycle counter filtering.
CCFVAL	Cycle Counter Filtering Value —This field defines the filter value for the cycle counter filtering.

Table 1061. Channel assignment description

CHA	CHB	Transmit message buffer		Receive message buffer	
		Static segment	Dynamic segment	Static segment	Dynamic segment
1	1	transmit on both channel A and channel B	transmit on channel A only	store first valid frame received on either channel A or channel B	store first valid frame received on channel A, ignore channel B
0	1	transmit on channel B	transmit on channel B	store first valid frame received on channel B	store first valid frame received on channel B
1	0	transmit on channel A	transmit on channel A	store first valid frame received on channel A	store first valid frame received on channel A
0	0	no frame transmission	no frame transmission	no frame stored	no frame stored

Note: *If at least one message buffer assigned to a certain slot is assigned to both channels, then all message buffers assigned to this slot have to be assigned to both channels. Otherwise, the message buffer configuration is illegal and the result of the message buffer search is not defined.*

51.5.2.81 Message Buffer Frame ID Registers (FR_MBFIDRn)

Figure 1050. Message Buffer Frame ID Registers (FR_MBFIDRn)

Base + 0x0804 (FR_MBFIDR0)

Base + 0x080C (FR_MBFIDR1)

16-bit write access required

Write: *POC:config* or MB_DIS

...

Base + 0x0BFC (FR_MBFIDR127)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	FID										
W																
Reset	0	0	0	0	0	—	—	—	—	—	—	—	—	—	—	—

Table 1062. FR_MBFIDRn field descriptions

Field	Description
FID	Frame ID —The semantic of this field depends on the message buffer transfer type. – <i>Receive Message Buffer</i>: This field is used as a filter value to determine if the message buffer is used for reception of a message received in a slot with the slot ID equal to FID. – <i>Transmit Message Buffer</i>: This field is used to determine the slot in which the message in this message buffer should be transmitted.

51.5.2.82 Message Buffer Index Registers (FR_MBIDXRn)

Figure 1051. Message Buffer Index Registers (FR_MBIDXRn)

Base + 0x0806 (FR_MBIDXR0)

Base + 0x080E (FR_MBIDXR1)

16-bit write access required

Write: *POC:config* or MB_DIS

...

Base + 0x0BFE (FR_MBIDXR127)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	MBIDX							
W																
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Table 1063. FR_MBIDXn field descriptions

Field	Description
MBIDX	Message Buffer Index —This field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer. The application writes the index of the initially associated message buffer header field into this register. The CC updates this register after frame reception or transmission. Legal Values are $0 \leq i \leq 131$. Illegal values will be detected during the message buffer search.

51.5.2.83 Message Buffer Data Field Offset Registers (FR_MBDORn)

Figure 1052. Message Buffer Data Field Offset Registers (FR_MBDORn)

Base + 0x1000 (FR_MBDOR0)

Base + 0x1002 (FR_MBDOR1)

16-bit write access required

Write: *POC:config*

...

Base + 0x1106 (FR_MBDOR131)

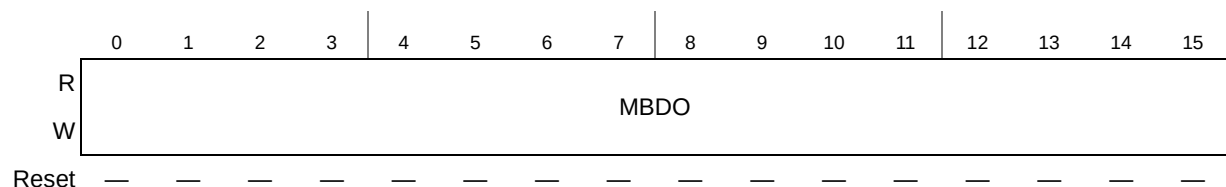


Table 1064. FR_MBDORn field descriptions

Field	Description
MBDO	Message Buffer Data Field Offset —This field provides the data field offset belonging to a particular Message Buffer Index. For configuration constraints see Section 51.7.1.2, "Configure data field offsets" .

51.5.2.84 LRAM ECC Error Test Registers (FR_LEETRn)

Figure 1053. LRAM ECC Error Test Registers (FR_LEETRn)

Base + 0x1108 (FR_LEETR0)

...

16-bit write access required

Write: *Anytime*

Base + 0x1112 (FR_LEETR5)

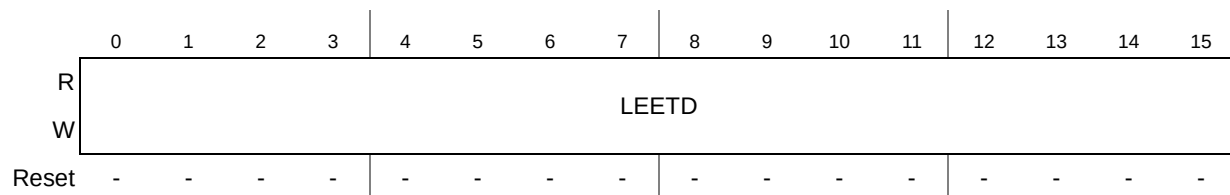


Table 1065. FR_LEETRn field descriptions

Field	Description
LEETD	LRAM ECC Error Test Data —This field contains the LRAM data belonging to the test register located in LRAM Bank n.

51.5.2.85 Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)

Figure 1054. Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)

Base + 0x00E2

Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	STP W _EN	0	0	0	0	0	TIM2 _EE	TIM1 _EE	CYS _EE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines whether or not the event output ports are enabled based on the events such as Stop watch count Cycle start and Timer1 and Timer2 expiry (see [Section 51.6.17, Timer support](#) for details).

Table 1066. FR_PEOER field descriptions

Field	Description
STPW_EN	Stop watch count Enable —This bit controls the stopwatch counter. 0 Stopwatch counter disabled 1 Stopwatch counter enabled
TIM2_EE	Timer 2 expired Event Output Enable —This bit controls the event on port “fr_evt_tim2” 0 Timer 2 expired event out disabled 1 Timer 2 expired event out enabled
TIM1_EE	Timer 1 expired Event Output Enable —This bit controls the event on port “fr_evt_tim1” 0 Timer 1 expired event out disabled 1 Timer 1 expired event out enabled
CYC_EE	Cycle Start Event Output Enable —This bit controls the event on port “fr_evt_cyc” 0 Cycle start event out disabled 1 Cycle start event out enabled

51.5.2.86 Stop Watch Count Register (FR_STPWR)

Figure 1055. Stop Watch Count High Register (FR_STPWHR)

Base + 0x000DE

Read Reset Only: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STPW{31:16}															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1056. Stop Watch Count Low Register (FR_STPWLR)

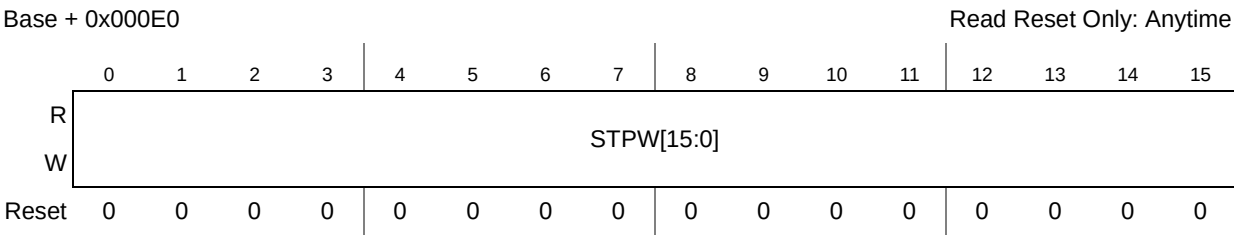


Table 1067. FR_STPWR field descriptions

Field	Description
STPW	Stop Watch Count Register —This is the value of the stop watch counter running on system clock. This starts on cycle start even and stops on an external event “OS_tick”. For details see Section 51.6.26, “StopWatch function . This is a read reset type register, which means that when all the 32 bits are read, it automatically clears.



51.6 Functional description

This section provides a detailed description of the functionality implemented in the CC.

51.6.1 Message buffer concept

The CC uses a data structure called a *message buffer* to store frame data, configuration, control, and status data. Each message buffer consists of two parts: the *message buffer control data* and the *physical message buffer*. The message buffer control data is located in dedicated registers. The structure of the message buffer control data depends on the message buffer type, described in [Section 51.6.3, "Message Buffer Types"](#). The physical message buffer is located in the FlexRay memory area and is described in [Section 51.6.2, "Physical message buffer"](#).

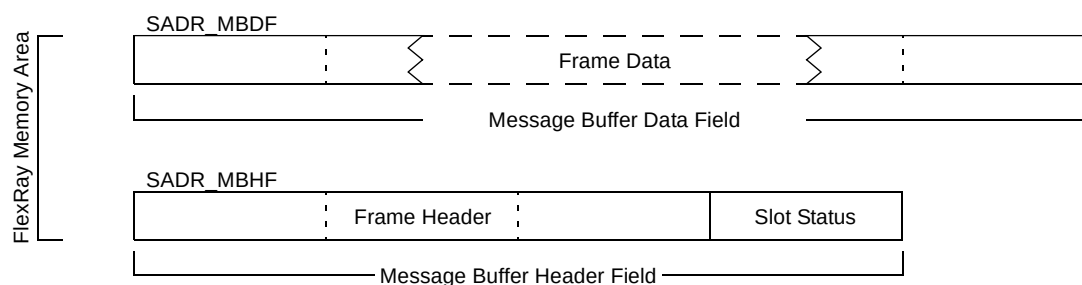
51.6.2 Physical message buffer

All FlexRay messages and related frame and slot status information of received frames and of frames to be transmitted to the FlexRay bus are stored in data structures called *physical message buffers*. The physical message buffers are located in the FlexRay memory area. The structure of a physical message buffer is depicted in [Figure 1053](#).

A physical message buffer consists of two fields, the *message buffer header field* and the *message buffer data field*. The message buffer header field contains the *frame header* and the *slot status*. The message buffer data field contains the *frame data*.

The connection between the two fields is established by the *data field offset*.

Figure 1057. Physical message buffer structure



51.6.2.1 Message buffer header field

The message buffer header field is a contiguous region in the FlexRay memory area that occupies eight bytes. It contains the frame header and the slot status. Its structure is shown in [Figure 1057](#). The physical start address SADR_MBHF of the message buffer header field must be 16-bit aligned.

51.6.2.1.1 Frame header

The frame header occupies the first six bytes in the message buffer header field. It contains all FlexRay frame header-related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the usage and the content of the frame header is provided in [Section 51.6.5.2.1, "Frame header description"](#).

51.6.2.1.2 Slot status

The slot status occupies the last 2 bytes of the message buffer header field. It provides the slot and frame status-related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the content and usage of the slot status is provided in [Section 51.6.5.2.2, “Slot status description](#).

51.6.2.2 Message buffer data field

The message buffer data field is a contiguous area of 2-byte entities. This field contains the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum length of this field depends on the specific message buffer configuration and is specified in the message buffer descriptions given in [Section 51.6.3, “Message Buffer Types](#).

51.6.3 Message Buffer Types

The CC provides three different types of message buffers.

- Individual Message Buffers
- Receive Shadow Buffers
- Receive FIFO Buffers

For each message buffer type, the structure of the physical message buffer is identical. The message buffer types differ only in the structure and content of message buffer control data, which control the related physical message buffer. The message buffer control data are described in the following sections.

51.6.3.1 Individual Message Buffers

The individual message buffers are used for all types of frame transmission and for dedicated frame reception based on individual filter settings for each message buffer. The CC supports three types of individual message buffers, which are described in [Section 51.6.6, “Individual message buffer functional description](#).

Each individual message buffer consists of two parts, the physical message buffer, which is located in the FlexRay memory area, and the message buffer control data, which are located in dedicated registers. The structure of an individual message buffer is given in [Figure 1058](#).

Each individual message buffer has a message buffer number n assigned, which determines the set of message buffer control registers associated to this individual message buffer. The individual message buffer with message buffer number n is controlled by the registers FR_MBCCSR n , FR_MBCCFR n , FR_MBFIDR n , and FR_MBIDXR n .

The connection between the message buffer control registers and the physical message buffer is established by the message buffer index field, MBIDX, in the [Message Buffer Index Registers \(FR_MBIDXR \$n\$ \)](#). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to [Equation 39](#).

$$\text{Equation 39} \quad \text{SADR_MBHF} = (\text{FR_MBIDXR}[\text{MBIDX}] \times 8) + \text{SMBA}$$

The data field belonging to a particular physical message buffer is characterized by the data field offset. For each physical message buffer with MBIDX i , the FR_MBDOR i contains the offset of the corresponding message buffer data field with respect to the CC FlexRay

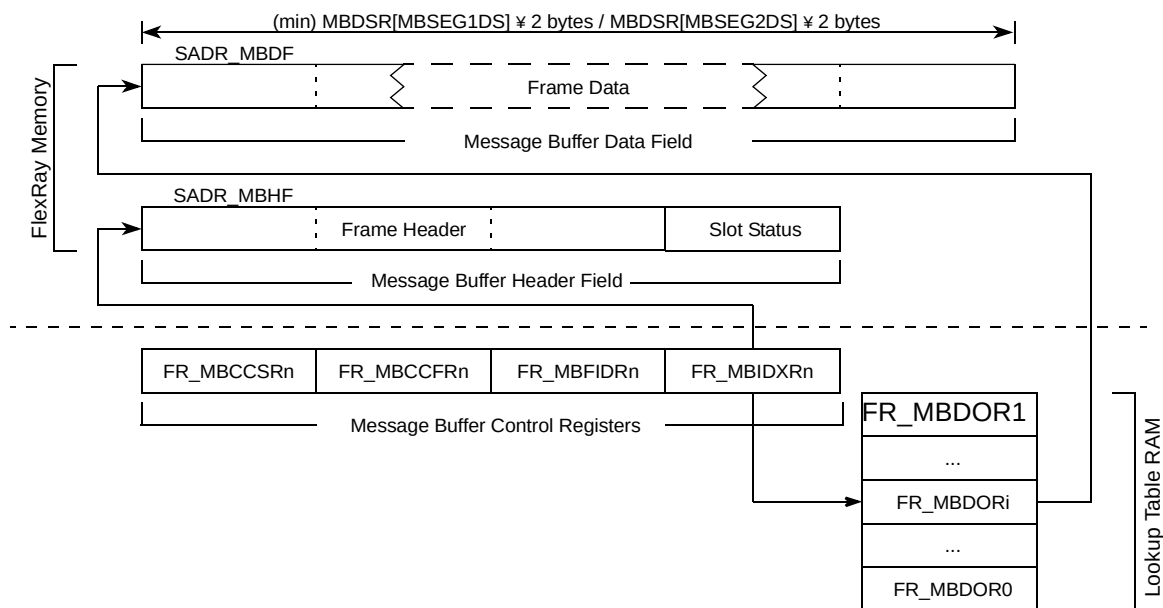
memory area base address as provided by SMBA field in the [System Memory Base Address Register \(FR_SYMBADR\)](#).

The data field offset is used to determine the start address SADR_MBDF of the corresponding message buffer data field in the FlexRay memory area according to [Equation 40](#).

$$\text{Equation 40} \quad \text{SADR_MBDF} = [\text{Data Field Offset}] + \text{SMBA}$$

The FR_MBDORn are stored in the module internal memory LRAM. Refer to [Section 51.7.2.3, "CHI LRAM initialization"](#) for the setup of the data field offset values.

Figure 1058. Individual message buffer structure



51.6.3.1.1 Individual message buffer segments

The set of the individual message buffers can be split up into two message buffer segments using the [Message Buffer Segment Size and Utilization Register \(FR_MBSSUTR\)](#). All individual message buffers with a message buffer number $n \leq \text{FR_MBSSUTR}[\text{LAST_MB_SEG1}]$ belong to the first message buffer segment. All individual message buffers with a message buffer number $n > \text{FR_MBSSUTR}[\text{LAST_MB_SEG1}]$ belong to the second message buffer segment. The following rules apply to the length of the message buffer data field:

- All physical message buffers associated to individual message buffers that belong to the same message buffer segment must have message buffer data fields of the same length.
- The minimum length of the message buffer data field for individual message buffers in the first message buffer segment is $2 \times \text{FR_MBDSR}[\text{MBSEG1DS}]$ bytes.
- The minimum length of the message buffer data field for individual message buffers assigned to the second segment is $2 \times \text{FR_MBDSR}[\text{MBSEG2DS}]$ bytes..

51.6.3.2 Receive shadow buffers

The receive shadow buffers are required for the individual message buffers' frame reception process. The CC provides four receive shadow buffers: one receive shadow buffer per channel and per message buffer segment.

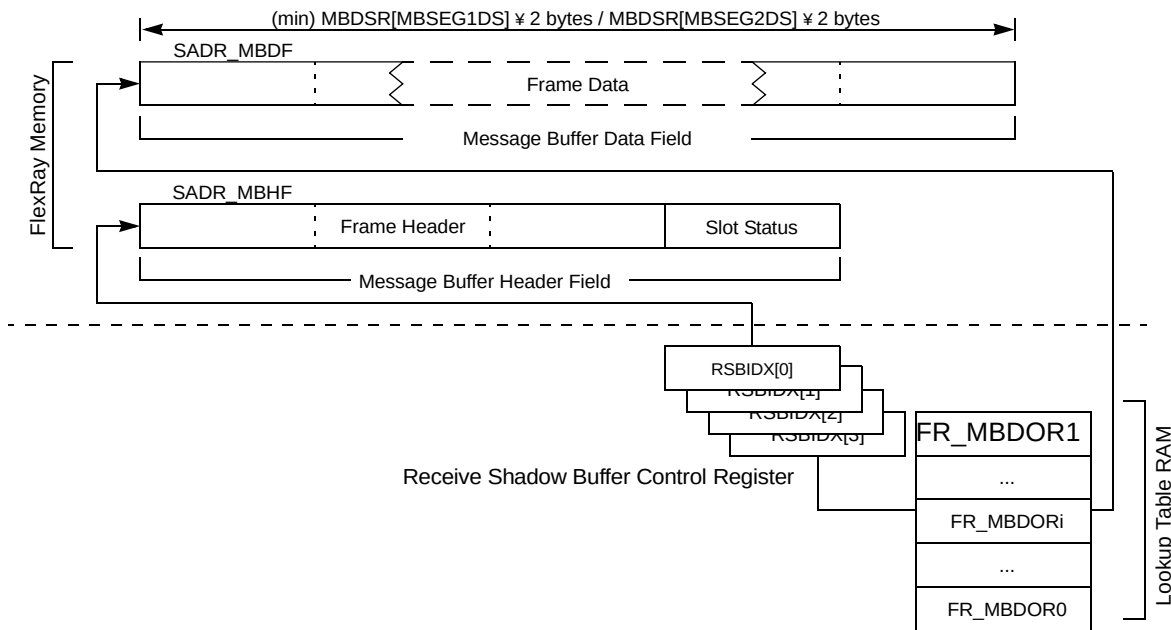
Each receive shadow buffer consists of two parts: the physical message buffer located in the FlexRay memory area and the receive shadow buffer control registers located in dedicated registers. The structure of a receive shadow buffer is shown in [Figure 1059](#). The four internal shadow buffer control registers can be accessed by the [Receive Shadow Buffer Index Register \(FR_RSBIR\)](#).

The connection between the receive shadow buffer control register and the physical message buffer for the selected receive shadow buffer is established by the receive shadow buffer index field RSBIDX in the [Receive Shadow Buffer Index Register \(FR_RSBIR\)](#). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to [Equation 41](#).

Equation 41 $SADR_MBHF = (FR_RSBIR[RSBIDX] \times 8) + SMBA$

The length required for the message buffer data field depends on the message buffer segment that the receive shadow buffer is assigned to. For the receive shadow buffers assigned to the first message buffer segment, the length must be the same as for the individual message buffers assigned to the first message buffer segment. For the receive shadow buffers assigned to the second message buffer segment, the length must be the same as for the individual message buffers assigned to the second message buffer segment. The receive shadow buffer assignment is described in [Receive Shadow Buffer Index Register \(FR_RSBIR\)](#).

Figure 1059. Receive shadow buffer structure



51.6.3.3 Receive FIFO

The receive FIFO implements a frame reception system based on the FIFO concept. The CC provides two independent receive FIFOs, one per channel.

A receive FIFO consists of a set of physical message buffers in the FlexRay memory area and a set of receive FIFO control registers located in dedicated registers. The structure of a receive FIFO is given in [Figure 1060](#).

The connection between the receive FIFO control registers and the set of physical message buffers is established by the [Receive FIFO Start Index Register \(FR_RFSIR\)](#), the [Receive FIFO Depth and Size Register \(FR_RFDSR\)](#), and the [Receive FIFO A Read Index Register \(FR_RFARIR\)/Receive FIFO B Read Index Register \(FR_RFBRIR\)](#).

The system memory base address SMBA valid for the receive FIFOs is defined by the system memory base address register selected by the FIFO address mode bit FR_MCR[FAM], refer to [Section 51.5.2.4, "Module Configuration Register \(FR_MCR\)"](#).

The start byte address SADR_MBHF[1] of the first message buffer header field that belongs to the receive FIFO is determined according to [Equation 42](#).

$$\text{Equation 42} \quad \text{SADR_MBHF}[1] = (8 \times \text{FR_RFSIR}[\text{SIDX}]) + \text{SMBA}$$

The start byte address SADR_MBHF[n] of the last message buffer header field that belongs to the receive FIFO in the FlexRay memory area is determined according to [Equation 43](#).

$$\text{Equation 43} \quad \text{SADR_MBHF}[n] = (8 \times (\text{FR_RFSIR}[\text{SIDX}] + \text{FR_RFDSR}[\text{FIFO_DEPTH}])) + \text{SMBA}$$

The required information to access the current entry of the FIFO is given in the following registers:

- The registers [Receive FIFO A Read Index Register \(FR_RFARIR\)](#) and [Receive FIFO B Read Index Register \(FR_RFBRIR\)](#) provide the index of the physical message buffer belonging to the current entry.

The data field offset belonging to the current FIFO entry RF_DFO[X] must be calculated using the current read index i according to the following formula:

$$\text{Equation 44} \quad \text{RF_DFO}[X] = \text{FR_RFSDOR}[X] + (\text{FR_RFDSR}[X][\text{ENTRY_SIZE}] \times 2) \times i - \text{FR_RFSIDX}[X]$$

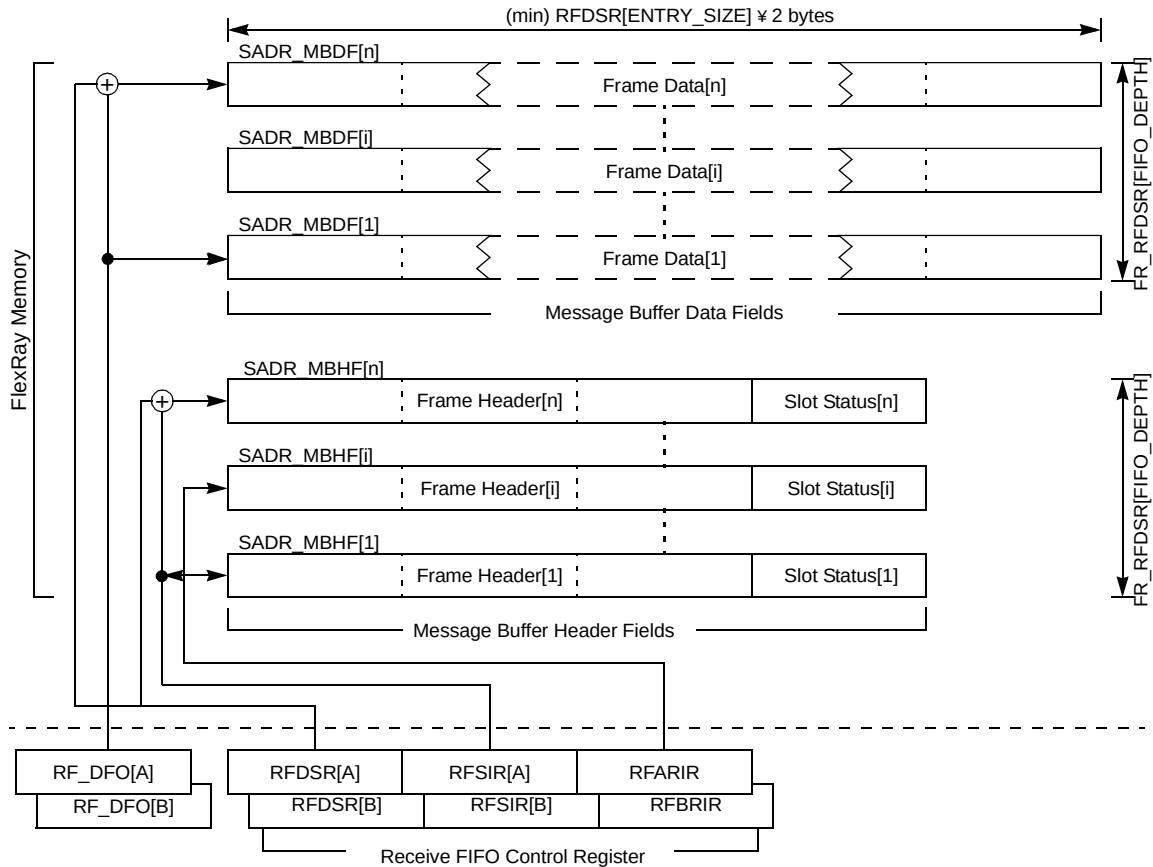
Note: The current read index loops up starting at the number given in the FR_RD[A/B]RDIDX register for the required number of entries.

Refer to [Section 51.6.9.8, "FIFO update"](#) for details about updating the FIFO read pointer.

All message buffer header fields assigned to a receive FIFO are within a contiguous region defined by FR_RFSIR[SIDX] and FR_RFDSR[FIFO_DEPTH].

The data sections of all FIFO entries within on receive FIFO are of the same length defined by FR_RFDSR[FIFO_SIZE].

Figure 1060. Receive FIFO structure



Note: The actual values of the data field offsets $RF_DFO[A/B]$ need to be calculated according to Equation 44. They are not stored in a register.

51.6.3.4 Message buffer configuration and control data

This section describes the configuration and control data for each message buffer type.

51.6.3.4.1 Individual message buffer configuration data

Before an individual message buffer can be used for transmission or reception, it must be configured. There is a set of common configuration parameters that applies to all individual message buffers and a set of configuration parameters that applies to each message buffer individually.

51.6.3.4.1.1 Common configuration data

The set of common configuration data for individual message buffers is located in the following registers.

- **Message Buffer Data Size Register (FR_MBDSR)**
The MBSEG2DS and MBSEG1DS fields define the minimum length of the message buffer data field with respect to the message buffer segment.
- **Message Buffer Segment Size and Utilization Register ($FR_MBSSUTR$)**
The LAST_MB_SEG1 and LAST_MB_UTIL fields define the segmentation of the individual message buffers and the number of individual message buffers that are

used. For more details, see [Section 51.6.3.1.1, “Individual message buffer segments](#).

51.6.3.4.1.2 Specific configuration data

The set of message buffer specific configuration data for individual message buffers is located in the following registers.

- [Message Buffer Configuration, Control, Status Registers \(FR_MBCCSRn\)](#)
The MTD bit configures the message buffer type.
- [Message Buffer Cycle Counter Filter Registers \(FR_MBCCFRn\)](#)
The MTM, CHA, CHB bits configure the transmission mode and the channel assignment. The CCFE, CCFMSK, and CCFVAL bits and fields configure the cycle counter filter.
- [Message Buffer Frame ID Registers \(FR_MBFIDRn\)](#)
For a transmit message buffer, the FID field is used to determine the slot in which the message in this message buffer will be transmitted.
- [Message Buffer Index Registers \(FR_MBIDXRn\)](#)
This MBIDX field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer.

51.6.3.5 Individual message buffer control data

During normal operation, each individual message buffer can be controlled by the CMT, LCKT, EDT, and MBIE control and trigger bits in the [Message Buffer Configuration, Control, Status Registers \(FR_MBCCSRn\)](#).

51.6.3.6 Receive shadow buffer configuration data

Before the individual message buffers can receive frames, the receive shadow buffers must be configured. The configuration data are provided by the [Receive Shadow Buffer Index Register \(FR_RSBIR\)](#). For each receive shadow buffer, the application provides the message buffer header index. When the protocol is in the *POC:normal active* or *POC:normal passive* state, the receive shadow buffers are under full CC control.

51.6.3.7 Receive FIFO control and configuration data

This section describes the configuration and control data for the two receive FIFOs.

51.6.3.7.1 Receive FIFO configuration data

The CC provides two functional independent receive FIFOs, one per channel. The FIFOs have a common subset of configuration data:

- [Receive FIFO Periodic Timer Register \(FR_RFPTR\)](#)

Each FIFO has its own set of configuration data. The configuration data are located in the following registers:

- Receive FIFO Watermark and Selection Register (FR_RFWMSR)
- Receive FIFO Start Index Register (FR_RFSIR)
- Receive FIFO Start Data Offset Register (FR_RFSDOR)
- Receive FIFO Depth and Size Register (FR_RFDSR)
- Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
- Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
- Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
- Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
- Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

51.6.3.7.2 Receive FIFO control data

The application can access the FIFOs at any time using the control bits in the following registers:

- Global Interrupt Flag and Enable Register (FR_GIFER)
- Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

51.6.3.7.3 Receive FIFO status data

The current status of the receive FIFO is provided in the following registers:

- Global Interrupt Flag and Enable Register (FR_GIFER)
- Receive FIFO A Read Index Register (FR_RFARIR)
- Receive FIFO B Read Index Register (FR_RFBRIR)
- Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

51.6.4 FlexRay memory area layout

The CC supports a wide range of possible layouts for the FlexRay memory area. Two basic layout modes can be selected by the FIFO address mode bit FR_MCR[FAM].

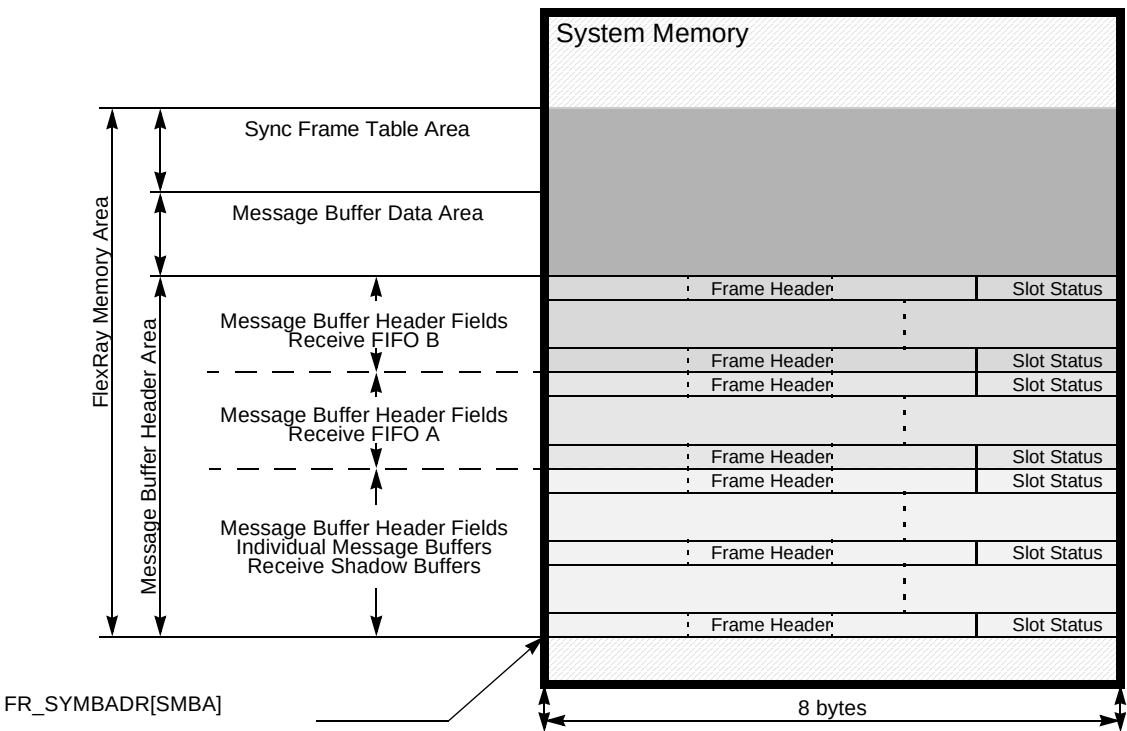
51.6.4.1 FlexRay memory area layout (FR_MCR[FAM] = 0)

[Figure 1061](#) shows an example layout for the FIFO address mode FR_MCR[FAM] = 0. In this mode, the following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area is one contiguous region.
- The FlexRay memory area size is a maximum of 64 KB.
- The FlexRay memory area starts at a 16 byte boundary.

The FlexRay memory area contains three areas: the *message buffer header area*, the *message buffer data area*, and the *sync frame table area*.

Figure 1061. Example of FlexRay memory area layout (FR_MCR[FAM] = 0)



51.6.4.2 FlexRay memory area layout (FR_MCR[FAM] = 1)

Figure 1062 shows an example layout for the FIFO address mode FR_MCR[FAM] = 1. The following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area consists of two contiguous regions.
- The size of each region is maximum 64 KB.
- Each region start at a 16 byte boundary.

$$\text{Equation 47} \quad \text{SADR_MBHF} = (i \times 8) + \text{FR_SYMBADR}[SMBA]; (0 \leq i \leq 1023)$$

3. The message buffer header fields for each FIFO must be a contiguous area.

51.6.4.4 Message buffer header area (FR_MCR[FAM] = 1)

The message buffer header area contains all message buffer header fields of the physical message buffers for the individual message buffers and receiver shadow buffers. The following rules apply to the message buffer header fields for the two type of message buffers.

1. The start address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill [Equation 48](#).

$$\text{Equation 48} \quad \text{SADR_MBHF} = (i \times 8) + \text{FR_SYMBADR}[SMBA]; (0 \leq i \leq 131)$$

51.6.4.5 FIFO message buffer header area (FR_MCR[FAM] = 1)

The FIFO message buffer header area contains all message buffer header fields of the physical message buffers for the FIFO. The following rules apply to the FIFO message buffer header fields.

1. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill [Equation 49](#).

$$\text{Equation 49} \quad \text{SADR_MBHF} = (i \times 8) + \text{FR_RFSYMBADR}[SMBA]; (0 \leq i \leq 1023)$$

2. The message buffer header fields for each FIFO have to be a contiguous area.

51.6.4.6 Message buffer data area

The message buffer data area contains all the message buffer data fields of the physical message buffers. Each message buffer data field must start at a 16-bit boundary.

51.6.4.7 Sync frame table area

The sync frame table area is used to provide a copy of the internal sync frame tables for application access. Refer to [Section 51.6.12, "Sync frame ID and sync frame deviation tables"](#) for the description of the sync frame table area.

51.6.5 Physical message buffer description

This section provides a detailed description of the usage and the content of the two parts of a physical message buffer: the message buffer header field and the message buffer data field.

51.6.5.1 Message buffer protection and data consistency

The physical message buffers are located in the FlexRay memory area. The CC provides no means to protect the FlexRay memory area from uncontrolled or illegal host or other client write access. To ensure data consistency of the physical message buffers, the application must follow the write access scheme that is given in the description of each of the physical message buffer fields.

51.6.5.2 Message buffer header field description

This section provides a detailed description of the usage and content of the message buffer header field. A description of the structure of the message buffer header fields is given in [Section 51.6.2.1, "Message buffer header field"](#). Each message buffer header field consists of two sections: the frame header section and the slot status section.

51.6.5.2.1 Frame header description

51.6.5.2.1.1 Frame header content

The semantic and content of the frame header section depends on the message buffer type.

For individual receive message buffers and receive FIFOs, the frame header receives the frame header data of the *first valid frame* received on the assigned channels.

For receive shadow buffers, the frame header receives the frame header data of the current frame received regardless of whether the frame is valid or not.

For transmit message buffers, the application writes the frame header of the frame to be transmitted into this location. The frame header will be read out when the frame is transferred to the FlexRay bus.

The structure of the frame header in the message buffer header field for receive message buffers and the receive FIFO is given in [Figure 1063](#). A detailed description is given in [Table 1069](#).

Figure 1063. Frame header structure (receive message buffer and receive FIFO)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID										
0x2	0	0	CYCCNT						0	PLDLEN						
0x4	0	0	0	0	0	HDCRC										

The structure of the frame header in the message buffer header field for transmit message buffers is given in [Figure 1064](#). A detailed description is given in [Table 1070](#). The checks that will be performed are described in [Frame Header Checks](#).

Figure 1064. Frame header structure (transmit message buffer)

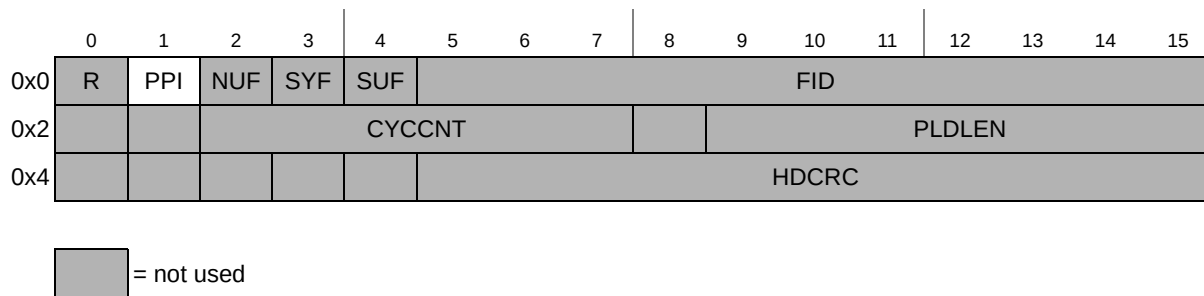
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0x0	R	PPI	NUF	SYF	SUF	FID											
0x2			CYCCNT								PLDLEN						
0x4						HDCRC											

 = not used

 = checked

 = checked if static slot

The structure of the frame header in the message buffer header field for transmit message buffers assigned to key slot is given in [Figure 1065](#).

Figure 1065. Frame header structure (transmit message buffer for key slot)

51.6.5.2.1.2 Frame Header Access

The frame header is located in the FlexRay memory area. To ensure data consistency, the application must follow the write access scheme described below.

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the frame header field.

For transmit message buffers, the application must follow the write access restrictions given in [Table 1068](#). This table shows the condition under which the application can write to the frame header entries without corrupting the FlexRay message transmission.

Table 1068. Frame header write access constraints (transmit message buffer)

Field	Static segment	Dynamic segment
FID	<i>POC:config</i> or MB_DIS	
PPI, PLDLEN, HDCRC	<i>POC:config</i> or MB_DIS or	MB_LCK

51.6.5.2.1.3 Frame Header Checks

As shown in [Figure 1064](#) and [Figure 1065](#), not all fields in the message buffer frame header are used for transmission. Some fields in the message buffer frame header are ignored, some are used for transmission, and some of them are checked for correct values. All checks that will be performed are described below.

For message buffers assigned to the key slot, no checks will be performed.

The value of the FID field must be equal to the value of the corresponding [Message Buffer Frame ID Registers \(FR_MBFIDRn\)](#). If the CC detects a mismatch while transmitting the frame header, it will set the frame ID error flag, FID_EF, in the [CHI Error Flag Register \(FR_CHIERFR\)](#). The value of the FID field will be ignored and replaced by the value provided in the [Message Buffer Frame ID Registers \(FR_MBFIDRn\)](#).

For transmit message buffers assigned to the *static* segment, the PLDLEN value must be equal to the value of the payload_length_static field in the [Protocol Configuration Register 19 \(FR_PCR19\)](#). If this is not fulfilled, the static payload length error flag SPL_EF in the [CHI Error Flag Register \(FR_CHIERFR\)](#) is set when the message buffer is under transmission. A syntactically and semantically correct frame is generated with payload_length_static payload words and the payload length field in the transmitted frame header set to payload_length_static.

For transmit message buffers assigned to the *dynamic* segment, the PLDLEN value must be less than or equal to the value of the `max_payload_length_dynamic` field in the [Protocol Configuration Register 24 \(FR_PCR24\)](#). If this is not fulfilled, the dynamic payload length error flag `DPL_EF` in the [CHI Error Flag Register \(FR_CHIERFR\)](#) is set when the message buffer is under transmission. A syntactically and semantically correct dynamic frame is generated with PLDLEN payload words and the payload length field in the frame header set to PLDLEN.

Table 1069. Frame header field descriptions (receive message buffer and receive FFO)

Field	Description
R	Reserved Bit —This is the value of the Reserved bit of the received frame stored in the message buffer.
PPI	Payload Preamble Indicator —This is the value of the Payload Preamble Indicator of the received frame stored in the message buffer.
NUF	Null Frame Indicator —This is the value of the Null Frame Indicator of the received frame stored in the message buffer.
SYF	Sync Frame Indicator —This is the value of the Sync Frame Indicator of the received frame stored in the message buffer.
SUF	Startup Frame Indicator —This is the value of the Startup Frame Indicator of the received frame stored in the message buffer.
FID	Frame ID —This is the value of the Frame ID field of the received frame stored in the message buffer.
CYCCNT	Cycle Count —This is the number of the communication cycle in which the frame stored in the message buffer was received.
PLDLEN	Payload Length —This is the value of the Payload Length field of the received frame stored in the message buffer.
HDCRC	Header CRC —This is the value of the Header CRC field of the received frame stored in the message buffer.

Table 1070. Frame header field descriptions (transmit message buffer)

Field	Description
R	Reserved Bit —This bit is not used. The value of the Reserved bit is generated internally according to <i>FlexRay Communications System Protocol Specification, Version 2.1 Rev A</i> .
PPI	Payload Preamble Indicator —This bit provides the value of the Payload Preamble Indicator for the frame transmitted from the message buffer.
NUF	Null Frame Indicator —This bit is not used, the value of the Null Frame Indicator is generated internally according to <i>FlexRay Communications System Protocol Specification, Version 2.1 Rev A</i> .
SYF	Sync Frame Indicator —This bit is not used. The value of the Sync Frame Indicator is generated internally according to <i>FlexRay Communications System Protocol Specification, Version 2.1 Rev A</i> .
SUF	Startup Frame Indicator —This bit is not used. The value of the Startup Frame Indicator is generated internally according to <i>FlexRay Communications System Protocol Specification, Version 2.1 Rev A</i> .
FID	Frame ID —This field is checked as described in Frame Header Checks .
CYCCNT	Cycle Count —This field is not used, the value of the transmitted Cycle Count field is taken from the internal communication cycle counter.

Table 1070. Frame header field descriptions (transmit message buffer)

Field	Description
PLDLEN	Payload Length —This field is checked and used as described in Frame Header Checks .
HDCRC	Header CRC —This field provides the value of the Header CRC field for the frame transmitted from the message buffer.

51.6.5.2.2 Slot status description

The slot status is a read-only structure for the application and a write-only structure for the CC. The meaning and content of the slot status in the message buffer header field depends on the message buffer type.

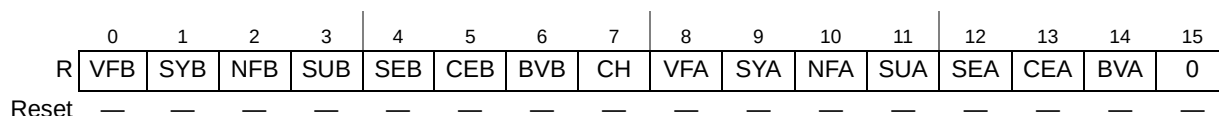
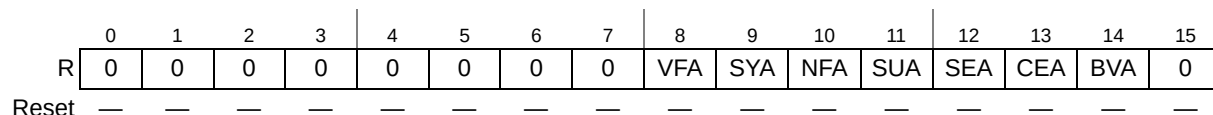
51.6.5.2.2.1 Receive message buffer and receive FIFO slot status description

This section describes the slot status structure for the individual receive message buffers and receive FIFOs. The content of the slot status structure for receive message buffers depends on the message buffer type and on the channel assignment for individual receive message buffers as given by [Table 1071](#).

Table 1071. Receive message buffer slot status content

Receive message buffer type	Slot status content
Individual Receive Message Buffer assigned to both channels FR_MBCCFRn[CHA] = 1 and FR_MBCCFRn[CHB] = 1	see Figure 1066
Individual Receive Message Buffer assigned to channel A FR_MBCCFRn[CHA] = 1 and FR_MBCCFRn[CHB] = 0	see Figure 1067
Individual Receive Message Buffer assigned to channel B FR_MBCCFRn[CHA] = 0 and FR_MBCCFRn[CHB] = 1	see Figure 1068
Receive FIFO Channel A Message Buffer	see Figure 1067
Receive FIFO Channel B Message Buffer	see Figure 1068

The meaning of the bits in the slot status structure is explained in [Table 1072](#).

**Figure 1066. Receive message buffer slot status structure (ChAB)****Figure 1067. Receive message buffer slot status structure (ChA)**

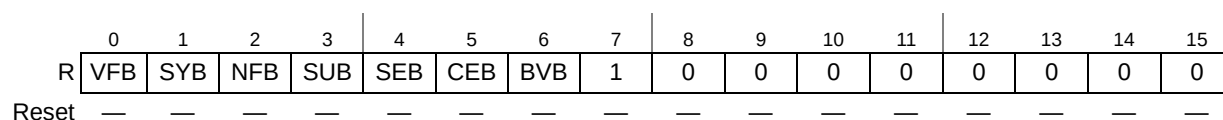


Figure 1068. Receive message buffer slot status structure (ChB)

Table 1072. Receive message buffer slot status field description

Field	Description
Common Message Buffer Status Bits	
VFB	Valid Frame on Channel B —Protocol-related variable: <i>vSS!ValidFrame</i> channel B 0 <i>vSS!ValidFrame</i> = 0 1 <i>vSS!ValidFrame</i> = 1
SYB	Sync Frame Indicator Channel B —Protocol-related variable: <i>vRF!Header!SyFIndicator</i> channel B 0 <i>vRF!Header!SyFIndicator</i> = 0 1 <i>vRF!Header!SyFIndicator</i> = 1
NFB	Null Frame Indicator Channel B —Protocol-related variable: <i>vRF!Header!NFIndicator</i> channel B 0 <i>vRF!Header!NFIndicator</i> = 0 1 <i>vRF!Header!NFIndicator</i> = 1
SUB	Startup Frame Indicator Channel B —Protocol-related variable: <i>vRF!Header!SuFIndicator</i> channel B 0 <i>vRF!Header!SuFIndicator</i> = 0 1 <i>vRF!Header!SuFIndicator</i> = 1
SEB	Syntax Error on Channel B —Protocol-related variable: <i>vSS!SyntaxError</i> channel B 0 <i>vSS!SyntaxError</i> = 0 1 <i>vSS!SyntaxError</i> = 1
CEB	Content Error on Channel B —Protocol related variable: <i>vSS!ContentError</i> channel B 0 <i>vSS!ContentError</i> = 0 1 <i>vSS!ContentError</i> = 1
BVB	Boundary Violation on Channel B —Protocol-related variable: <i>vSS!BViolation</i> channel B 0 <i>vSS!BViolation</i> = 0 1 <i>vSS!BViolation</i> = 1
CH	Channel first valid received —This status bit applies only to receive message buffers assigned to the static segment and to both channels. It indicates the channel that has received the <i>first valid</i> frame in the slot. This flag is set to 0 if no valid frame was received at all in the subscribed slot. 0 First valid frame received on channel A, or no valid frame received at all 1 First valid frame received on channel B
VFA	Valid Frame on Channel A —Protocol-related variable: <i>vSS!ValidFrame</i> channel A 0 <i>vSS!ValidFrame</i> = 0 1 <i>vSS!ValidFrame</i> = 1
SYA	Sync Frame Indicator Channel A —Protocol-related variable: <i>vRF!Header!SyFIndicator</i> channel A 0 <i>vRF!Header!SyFIndicator</i> = 0 1 <i>vRF!Header!SyFIndicator</i> = 1
NFA	Null Frame Indicator Channel A —Protocol-related variable: <i>vRF!Header!NFIndicator</i> channel A 0 <i>vRF!Header!NFIndicator</i> = 0 1 <i>vRF!Header!NFIndicator</i> = 1

Table 1072. Receive message buffer slot status field description (continued)

Field	Description
SUA	Startup Frame Indicator Channel A —Protocol-related variable: <i>vRF!Header!SuFIndicator</i> channel A 0 <i>vRF!Header!SuFIndicator</i> = 0 1 <i>vRF!Header!SuFIndicator</i> = 1
SEA	Syntax Error on Channel A —Protocol-related variable: <i>vSS!SyntaxError</i> channel A 0 <i>vSS!SyntaxError</i> = 0 1 <i>vSS!SyntaxError</i> = 1
CEA	Content Error on Channel A —Protocol-related variable: <i>vSS!ContentError</i> channel A 0 <i>vSS!ContentError</i> = 0 1 <i>vSS!ContentError</i> = 1
BVA	Boundary Violation on Channel A —Protocol-related variable: <i>vSS!BViolation</i> channel A 0 <i>vSS!BViolation</i> = 0 1 <i>vSS!BViolation</i> = 1

51.6.5.2.2.2 Transmit Message Buffer Slot Status Description

This section describes the slot status structure for transmit message buffers. Only the TCA and TCB status bits are directly related to the transmission process. All other status bits in this structure are related to a receive process that may have occurred. The content of the slot status structure for transmit message buffers depends on the channel assignment as given by [Table 1073](#).

Table 1073. Transmit message buffer slot status content

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Figure 1069
Individual Transmit Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Figure 1070
Individual Transmit Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Figure 1071

The meaning of the bits in the slot status structure is described in [Table 1074](#).

Figure 1069. Transmit message buffer slot status structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 1070. Transmit message buffer slot status structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 1071. Transmit message buffer slot status structure (ChB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	0	0	0	0	0	0	0	0
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 1074. Transmit message buffer slot status structure field description

Field	Description
VFB	Valid Frame on Channel B —Protocol-related variable: <i>vSS!ValidFrame</i> channel B 0 <i>vSS!ValidFrame</i> = 0 1 <i>vSS!ValidFrame</i> = 1
SYB	Sync Frame Indicator Channel B —Protocol-related variable: <i>vRF!Header!SyFIndicator</i> channel B 0 <i>vRF!Header!SyFIndicator</i> = 0 1 <i>vRF!Header!SyFIndicator</i> = 1
NFB	Null Frame Indicator Channel B —Protocol-related variable: <i>vRF!Header!NFIndicator</i> channel B 0 <i>vRF!Header!NFIndicator</i> = 0 1 <i>vRF!Header!NFIndicator</i> = 1
SUB	Startup Frame Indicator Channel B —Protocol-related variable: <i>vRF!Header!SuFIndicator</i> channel B 0 <i>vRF!Header!SuFIndicator</i> = 0 1 <i>vRF!Header!SuFIndicator</i> = 1
SEB	Syntax Error on Channel B —Protocol-related variable: <i>vSS!SyntaxError</i> channel B 0 <i>vSS!SyntaxError</i> = 0 1 <i>vSS!SyntaxError</i> = 1
CEB	Content Error on Channel B —Protocol-related variable: <i>vSS!ContentError</i> channel B 0 <i>vSS!ContentError</i> = 0 1 <i>vSS!ContentError</i> = 1
BVB	Boundary Violation on Channel B —Protocol-related variable: <i>vSS!BViolation</i> channel B 0 <i>vSS!BViolation</i> = 0 1 <i>vSS!BViolation</i> = 1
TCB	Transmission Conflict on Channel B —Protocol-related variable: <i>vSS!TxConflict</i> channel B 0 <i>vSS!TxConflict</i> = 0 1 <i>vSS!TxConflict</i> = 1
VFA	Valid Frame on Channel A —Protocol-related variable: <i>vSS!ValidFrame</i> channel A 0 <i>vSS!ValidFrame</i> = 0 1 <i>vSS!ValidFrame</i> = 1
SYA	Sync Frame Indicator Channel A —Protocol-related variable: <i>vRF!Header!SyFIndicator</i> channel A 0 <i>vRF!Header!SyFIndicator</i> = 0 1 <i>vRF!Header!SyFIndicator</i> = 1
NFA	Null Frame Indicator Channel A —Protocol-related variable: <i>vRF!Header!NFIndicator</i> channel A 0 <i>vRF!Header!NFIndicator</i> = 0 1 <i>vRF!Header!NFIndicator</i> = 1
SUA	Startup Frame Indicator Channel A —Protocol-related variable: <i>vRF!Header!SuFIndicator</i> channel A 0 <i>vRF!Header!SuFIndicator</i> = 0 1 <i>vRF!Header!SuFIndicator</i> = 1

Table 1074. Transmit message buffer slot status structure field description

Field	Description
SEA	Syntax Error on Channel A —Protocol-related variable: <i>vSS!SyntaxError</i> channel A 0 <i>vSS!SyntaxError</i> = 0 1 <i>vSS!SyntaxError</i> = 1
CEA	Content Error on Channel A —Protocol-related variable: <i>vSS!ContentError</i> channel A 0 <i>vSS!ContentError</i> = 0 1 <i>vSS!ContentError</i> = 1
BVA	Boundary Violation on Channel A —Protocol-related variable: <i>vSS!BViolation</i> channel A 0 <i>vSS!BViolation</i> = 0 1 <i>vSS!BViolation</i> = 1
TCA	Transmission Conflict on Channel A —Protocol-related variable: <i>vSS!TxConflict</i> channel A 0 <i>vSS!TxConflict</i> = 0 1 <i>vSS!TxConflict</i> = 1

51.6.5.3 Message buffer data field description

The message buffer data field is used to store the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum required length of this field depends on the message buffer type that the physical message buffer is assigned to and is given in [Table 1075](#). The structure of the message buffer data field is given in [Figure 1072](#).

Table 1075. Message buffer data field minimum length

Physical message buffer assigned to	Minimum length defined by
Individual message buffer in segment 1	FR_MBDSR[MBSEG1DS]
Receive shadow buffer in segment 1	FR_MBDSR[MBSEG1DS]
Individual message buffer in segment 2	FR_MBDSR[MBSEG2DS]
Receive shadow buffer in segment 2	FR_MBDSR[MBSEG2DS]
Receive FIFO for channel A	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 0)
Receive FIFO for channel B	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 1)

Note: The CC will not access any locations outside the message buffer data field boundaries given by [Table 1075](#).

Figure 1072. Message buffer data field structure

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	DATA0 / MID0 / NMV0								DATA1 / MID1 / NMV1							
0x2	DATA2 / NMV2								DATA3 / NMV3							
...	...								...							
0xN-2	DATA N-2								DATA N-1							

The message buffer data field is located in the FlexRay memory area; thus, the CC has no means to control application write access to the field. To ensure data consistency, the application must follow a write and read access scheme.

51.6.5.3.1 Message buffer data field read access

For transmit message buffers, the CC will not modify the content of the message buffer data field. Thus the application can read back the data at any time without any impact on data consistency.

For receive message buffers the application must lock the related receive message buffer and retrieve the message buffer header index from the [Message Buffer Index Registers \(FR_MBIDXn\)](#). While the message buffer is locked, the CC will not update the message buffer data field.

For receive FIFOs, the application can read the message buffer indicated by the [Receive FIFO A Read Index Register \(FR_RFARIR\)](#) or the [Receive FIFO B Read Index Register \(FR_RFBIR\)](#) when the related fill levels in the [Receive FIFO Fill Level and POP Count Register \(FR_RFFLPCR\)](#) indicate an non-empty FIFO.

51.6.5.3.2 Message buffer data field write access

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the message buffer data field.

For transmit message buffers, the application must follow the write access restrictions given in [Table 1076](#).

Table 1076. Frame data write access constraints

Field	CC/MB State
DATA, MID, NMV	<i>POC:config</i> or MB_DIS or MB_LCK

Table 1077. Frame data field descriptions

Field	Description
DATA 0, DATA 1, ... DATA N–1	Message Data —Provides the message data received or to be transmitted. For receive message buffer and receive FIFOs, this field provides the message data received for this message buffer. For transmit message buffers, the field provides the message data to be transmitted.
MID 0, MID 1	Message Identifier —If the payload preamble bit PPI is set in the message buffer frame header, the MID field holds the message ID of a dynamic frame located in the message buffer. The receive FIFO filter uses the received message ID for message ID filtering.
NMV 0, NMV 1, ... NMV 11	Network Management Vector —If the payload preamble bit PPI is set in the message buffer frame header, the network management vector field holds the network management vector of a static frame located in the message buffer. <i>Note: The MID and NMV bytes replace the corresponding DATA bytes.</i>

51.6.6 Individual message buffer functional description

The CC provides the following basic types of individual message buffers:

- Transmit message buffers
- Receive message buffers

Before an individual message buffer can be used, it must be configured by the application. After the initial configuration, the message buffer can be reconfigured later. The set of the configuration data for individual message buffers is given in [Section 51.6.3.4.1, "Individual message buffer configuration data"](#).

51.6.6.1 Individual message buffer configuration

The individual message buffer configuration consists of two steps. The first step is the allocation of the required amount of memory for the FlexRay memory area. The second step is the programming of the message buffer configuration registers, which is described in this section.

51.6.6.1.1 Common configuration data

One part of the message buffer configuration data is common to all individual message buffers and the receive shadow buffers. This data can only be set when the protocol is in the *POC:config* state.

The application configures the number of utilized individual message buffers by writing the message buffer number of the last utilized message buffer into the LAST_MB_UTIL field in the [Message Buffer Segment Size and Utilization Register \(FR_MBSSUTR\)](#).

The application configures the size of the two segments of individual message buffers by writing the message buffer number of the last message buffer in the first segment into the LAST_MB_SEG1 field in the [Message Buffer Segment Size and Utilization Register \(FR_MBSSUTR\)](#).

The application configures the length of the message buffer data fields for both of the message buffer segments by writing to the MBSEG2DS and MBSEG1DS fields in the [Message Buffer Data Size Register \(FR_MBDSSR\)](#).

Depending on the current receive functionality of the CC, the application must configure the receive shadow buffers. For each segment and for each channel with at least one individual receive message buffer assigned, the application must configure the related receive shadow buffer using the [Receive Shadow Buffer Index Register \(FR_RSBR\)](#).

51.6.6.1.2 Specific configuration data

The second part of the message buffer configuration data is specific for each message buffer.

This data can be changed only when either

- The protocol is in the *POC:config* state, or
- The message buffer is disabled (that is, FR_MBCCSRn[EDS] = 0)

The individual message buffer type is defined by the MTD and MBT bits in the [Message Buffer Configuration, Control, Status Registers \(FR_MBCCSRn\)](#) as given in [Table 1078](#).

Table 1078. Individual message buffer types

FR_MBCCSRn		Individual message buffer description
MTD	MBT	
0	0	Receive message buffer
0	1	Reserved
1	0	Transmit message buffer
1	1	Reserved

The message-buffer-specific configuration data includes the following:

1. MTD bits in the [Message Buffer Configuration, Control, Status Registers \(FR_MBCCSRn\)](#)
2. All fields in the [Message Buffer Cycle Counter Filter Registers \(FR_MBCCFRn\)](#)
3. All fields in the [Message Buffer Frame ID Registers \(FR_MBFIDRn\)](#)
4. All fields in the [Message Buffer Index Registers \(FR_MBIDXRn\)](#)

The meaning of the specific configuration data depends on the message buffer type, as given in the detailed message buffer type descriptions. See [Section 51.6.6.2, “Transmit message buffers”](#) and [Section 51.6.6.3, “Receive message buffers”](#) for further information.

51.6.6.2 Transmit message buffers

The section provides a detailed description of the functionality of single-buffered transmit message buffers.

A transmit message buffer is used by the application to provide message data to the CC that will be transmitted over the FlexRay Bus. The CC uses the transmit message buffers to provide information about the transmission process and status information about the slot in which message was transmitted.

The individual message buffer with message buffer number *n* is configured to be a transmit message buffer by the following settings:

- FR_MBCCSRn[MBT] = 0 (single buffered message buffer)
- FR_MBCCSRn[MTD] = 1 (transmit message buffer)

51.6.6.2.1 Access regions

For certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is used to control the access to the data, control, and status bits of a message buffer. The access regions for transmit message buffers are depicted in [Figure 1073](#). A description of the regions is given in [Table 1080](#). If a region is active as indicated in [Table 1081](#), the access scheme given for that region applies to the message buffer.

Figure 1073. Transmit message buffer access regions

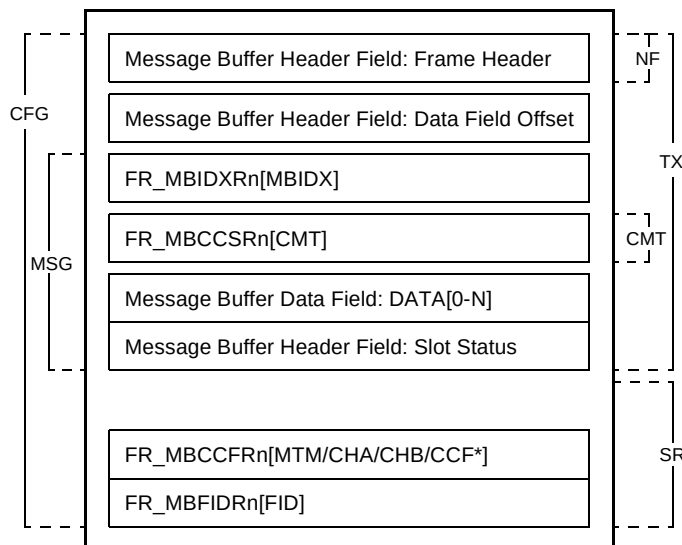


Table 1079. Transmit message buffer access regions description

Region	Access from		Region used for
	Application	Module	
CFG	Read/write	—	Message buffer configuration
MSG	Read/write	—	Message data and slot status access
NF	—	Read-only	Message header access for null frame transmission
TX	—	Read/write	Message transmission and slot status update
CM	—	Read-only	Message buffer validation
SR	—	Read-only	Message buffer search

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT], and the interrupt enable bit FR_MBCCSRn[MBIE], are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC clear access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The transmit message buffer states are given in [Figure 1074](#). A description of the states is given in [Table 1080](#), which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required message buffer status information. The internal status information is not visible to the application.

51.6.6.2.2 Message buffer states

This section describes the transmit message buffer states and provides a state diagram.


```

graph TD
    RESET_STATE[RESET_STATE] --> HDis
    HDis -- HD --> Idle
    Idle -- HE --> HDis
    Idle -- HL --> CCSa
    CCSa -- SA --> Idle
    Idle -- HU --> HDisLck
    HDisLck -- HU --> Idle
    HDisLck -- HL --> HLck
    HLck -- HL --> HLckCCSa
    HLckCCSa -- HL --> HLck
    HLck -- HU --> CCSa
    CCSa -- HU --> HLckCCSa
    HLck -- HD --> HDisLck
    HDisLck -- HD --> HLck
    HLck -- DSS --> HLckCCSa
    HLckCCSa -- DSS --> HLck
    HLck -- MA --> HLckCCnf
    HLckCCnf -- MA --> HLck
    HLckCCnf -- STS --> HLckCCMa
    HLckCCMa -- STS --> HLckCCnf
    HLckCCnf -- HU --> CCnf
    CCnf -- HU --> HLckCCnf
    CCnf -- STS --> CCMa
    CCMa -- STS --> CCnf
    CCnf -- MA --> CCTx
    CCTx -- MA --> CCnf
    CCTx -- TX --> CCMa
    CCMa -- TX --> CCTx
    CCMa -- HU --> HLckCCMa
    HLckCCMa -- HU --> CCMa
    CCMa -- DSS --> CCSu
    CCSu -- DSS --> CCMa
    CCSu -- SU --> Idle
    Idle -- DSS --> CCSu
    HLckCCMa -- DSS --> HLckCCnf
    HLckCCnf -- DSS --> HLckCCMa
  
```

State	FR_MBCCSRn		Access region		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	—	CM, SR	Idle —Message Buffer is idle. Included in message buffer search
HDis	0	0	CFG	—	Disabled —Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	—	Disabled and Locked —Message Buffer under configuration. Excluded from message buffer search.
HLck	1	1	MSG	SR	Locked—Application access to data, control, and status. Included in message buffer search.
CCSa	1	0	—	—	Slot Assigned—Message buffer assigned to next static slot. Ready for Null Frame transmission.
HLckCCSa	1	1	MSG	—	Locked and Slot Assigned—Application access to data, control, and status.Message buffer assigned to next static slot.
CCNf	1	0	—	NF	Null Frame Transmission—Header is used for null frame transmission.
HLckCCNf	1	1	MSG	NF	Locked and Null Frame Transmission—Application access to data, control, and status. Header is used for null frame transmission.
CCMa	1	0	—	CM	Message Available—Message buffer is assigned to next slot and cycle counter filter matches.

Table 1080. Transmit message buffer state description (continued)

State	FR_MBCCSRn		Access region		Description
	EDS	LCKS	Appl.	Module	
HLckCCMa	1	1	MSG	—	Locked and Message Available—Applications access to data, control, and status. Message buffer is assigned to next slot and cycle counter filter matches.
CCTx	1	0	—	TX	Message Transmission—Message buffer data transmit. Payload data from buffer transmitted.
CCSu	1	0	—	TX	Status Update—Message buffer status update. Update of status flags, the slot status field, and the header index.

51.6.6.2.3 Message buffer transitions

51.6.6.2.3.1 Application transitions

The application transitions can be triggered by the application using the commands described in [Table 1081](#). The application issues the commands by writing to the [Message Buffer Configuration, Control, Status Registers \(FR_MBCCSRn\)](#). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

Message Buffer Enable and Disable

The enable and disable commands are issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in the CCSa, HLckCCSa, CCMa, HLckCCMa, CCNf, HLckCCNf, or CCTx state, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

Message Buffer Lock and Unlock

The lock and unlock commands are issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL and the message buffer is in the state CCTx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the [CHI Error Flag Register \(FR_CHIERFR\)](#) is set.

Table 1081. Transmit message buffer application transitions

Transition	Command	Condition	Description
HE	FR_MBCCSRn[EDT] := 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.

Table 1081. Transmit message buffer application transitions (continued)

Transition	Command	Condition	Description
HL	FR_MBCCSRn[LCKT] = 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

51.6.6.2.3.2 Module transitions

The module transitions that can be triggered by the CC are described in [Table 1082](#). Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 1082. Transmit message buffer module transitions

Transition	Condition	Description
SA	Slot match and static slot	Slot Assigned—Message buffer is assigned to next static slot.
MA	Slot match and CycleCounter match	Message Available—Message buffer is assigned to next slot and cycle counter filter matches.
TX	Slot start and FR_MBCCSRn[CMT] = 1	Transmission Slot Start—Slot Start and commit bit CMT is set. In case of a dynamic slot, pLatestTx is not exceeded.
SU	Status updated	Status Updated—Slot Status field and message buffer status flags updated. Interrupt flag set.
STS	Static slot start	Static Slot Start—Start of static slot.
DSS	Dynamic slot start or symbol window start or NIT start	Dynamic Slot or Segment Start—Start of dynamic slot or symbol window or NIT.
SSS	Slot start or symbol window start or NIT start	Slot or Segment Start—Start of static slot or dynamic slot or symbol window or NIT.

51.6.6.2.3.3 Transition priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the first part of [Table 1083](#), the module transitions have a higher priority than the application transitions. For all states except the CCMA state, both a lock/unlock transition HL/HD and a module transition can be executed at the same time. The result state is reached by first applying the application transition and, subsequently, the module transition to the intermediately reached state. For example, if the message buffer is in the HLck state and the application unlocks the message buffer by the HU transition and the module triggers the slot assigned transition SA, the intermediate state is Idle and the resulting state is CCSa.

The priorities among the module transitions are given in the second part of [Table 1083](#).

Table 1083. Transmit message buffer transition priorities

State	Priorities	Description
module vs. application		
Idle, HLck	SA > HD	Slot Assigned > Message Buffer Disable
	MA > HD	Message Available > Message Buffer Disable
CCMa	TX > HL	Transmission Start > Message Buffer Lock
Module internal		
Idle, HLck	MA > SA	Message Available > Slot Assigned
CCMa	TX > STS	Transmission Slot Start > Static Slot Start
	TX > DSS	Transmission Slot Start > Dynamic Slot Start

51.6.6.2.4 Transmit message setup

To transmit a message over the FlexRay bus, the application writes the message data into the message buffer data field and sets the commit bit, CMT, in the [Message Buffer Configuration, Control, Status Registers \(FR_MBCCSRn\)](#). The physical access to the message buffer data field is described in [Section 51.6.3.1, "Individual Message Buffers"](#).

As indicated by [Table 1074](#), the application will write to the message buffer data field and change the commit bit CMT only if the transmit message buffer is in one of the following states: HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa. The application can change the state of a message buffer if it issues the appropriate commands shown in [Table 1074](#). The state change is indicated through the FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] status bits.

If the transmit message buffer enters the HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa state, the FR_MBCCSRn[DVAL] flag is negated.

51.6.6.2.5 Message transmission

As a result of the message buffer search described in [Section 51.6.7, "Individual message buffer search"](#), the CC triggers the message available transition MA for up to two transmit message buffers. This changes the message buffer state from Idle to CCMa and the message buffers can be used for message transmission in the next slot.

The CC transmits a message from a message buffer if both of the following two conditions are fulfilled at the start of the transmission slot:

1. The message buffer is in the message available state CCMa
2. The message data is still valid (that is, FR_MBCCSRn[CMT] = 1)

In this case, the CC triggers the TX transition and changes the message buffer state to CCTx. A transmit message buffer timing and state change diagram for message transmission is given in [Figure 1075](#). In this example, the message buffer with message buffer number *n* is Idle at the start of the search slot, matches the slot and cycle number of the next slot, and message buffer data are valid (that is, FR_MBCCSRn[CMT] = 1).

Figure 1075. Message transmission timing

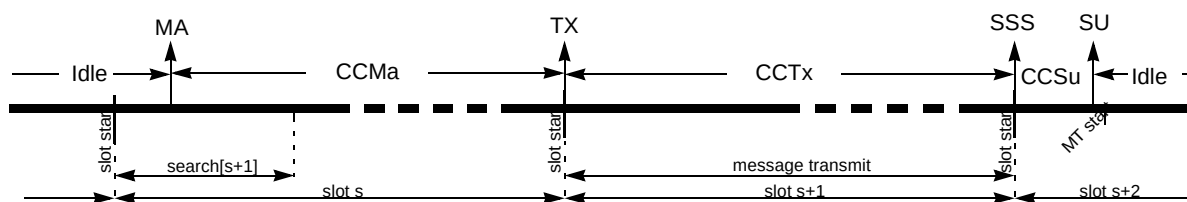
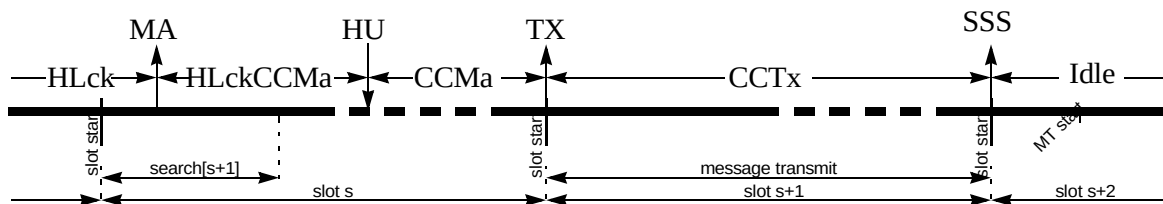


Figure 1076. Message transmission from HLck state with unlock



The amount of message data read from the FlexRay memory area and transferred to the FlexRay bus is determined by the following three items

1. The message buffer segment that the message buffer is assigned to, as defined by the [Message Buffer Segment Size and Utilization Register \(FR_MBSSUTR\)](#).
2. The message buffer data field size, as defined by the related field of the [Message Buffer Data Size Register \(FR_MBDSDR\)](#)
3. The value of the PLDLEN field in the message buffer header field, as described in [Section 51.6.5.2.1, "Frame header description"](#)

If a message buffer is assigned to message buffer segment 1, and $PLDLEN > MBSEG1DS$, then $2 \times MBSEG1DS$ bytes will be read from the message buffer data field and zero padding is used for the remaining bytes for the FlexRay bus transfer. If $PLDLEN \leq MBSEG1DS$, the CC reads and transfers $2 \times PLDLEN$ bytes. The same holds for segment 2 and $MBSEG2DS$.

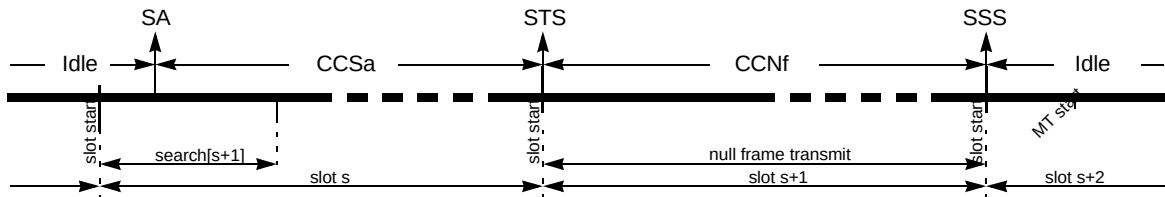
51.6.6.2.6 Null frame transmission

A static slot with slot number S is assigned to the CC for channel A, if at least one transmit message buffer is configured with the $FR_MBFIDRn[FID]$ set to S and $FR_MBCCFRn[CHA]$ set to 1. A null frame is transmitted in the static slot s on channel A, if this slot is assigned to the CC for channel A, and all transmit message buffers with $FR_MBFIDRn[FID] = S$ and $FR_MBCCFRn[CHA] = 1$ are either not committed (that is, $FR_MBCCSRn[CMT] = 0$), or are locked by the application (that is, $FR_MBCCSRn[LCKS] = 1$), or the cycle counter filter is enabled and does not match.

Additionally, the application can clear the commit bit of a message buffer that is in the CCMa state, which is called *uncommit* or *transmit abort*. This message buffer will be used for null frame transmission.

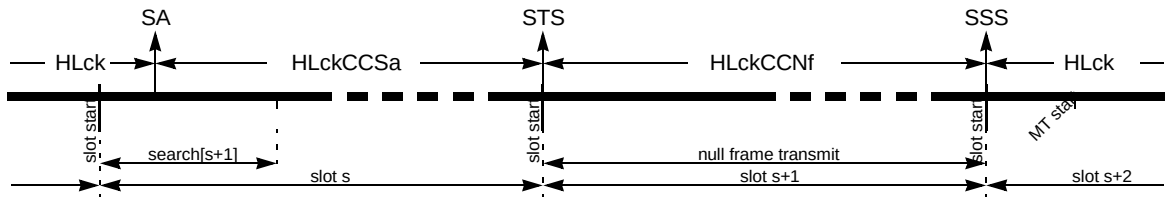
As a result of the message buffer search described in [Section 51.6.7, "Individual message buffer search"](#), the CC triggers the slot assigned transition SA for up to two transmit message buffers if at least one of the conditions mentioned above is fulfilled for these message buffers. The transition SA changes the message buffer states from either Idle to CCSa or from HLck to HLckCCSa. In each case, these message buffers will be used for null frame transmission in the next slot. A message buffer timing and state change diagram for null frame transmission from Idle state is given in [Figure 1077](#).

Figure 1077. Null frame transmission from Idle state



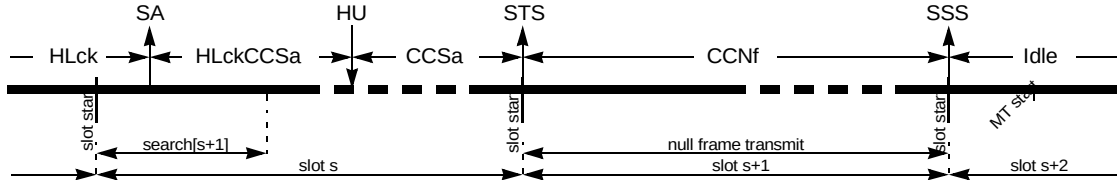
A message buffer timing and state change diagram for null frame transmission from HLck state is given in [Figure 1078](#).

Figure 1078. Null frame transmission from HLck state



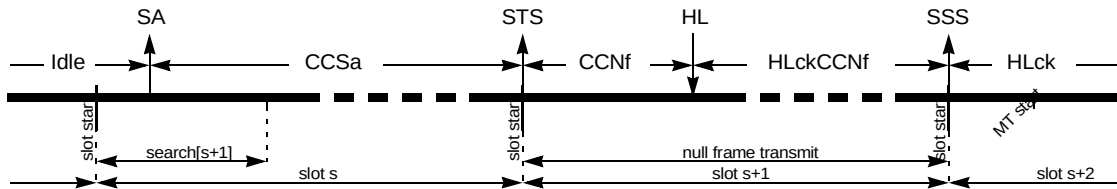
If a transmit message buffer is in the CCSa or HLckCCSa state at the start of the transmission slot, a null frame is transmitted in any case, even if the message buffer is unlocked or committed before the transmission slot starts. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in [Figure 1079](#).

Figure 1079. Null frame transmission from HLck state with unlock



Since the null frame transmission will not use the message buffer data, the application can lock/unlock the message buffer during null frame transmission. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in [Figure 1080](#).

Figure 1080. Null frame transmission from Idle state with locking



51.6.6.2.7 Message buffer status update

After the end of each slot, the PE generates the slot status vector. Depending on the status, the transmitted frame type, and the amount of transmitted data, the message buffer status is updated.

51.6.6.2.7.1 Message buffer status update after complete message transmission

The term complete message transmission refers to the fact that all payload data stored in the message buffer was sent to the FlexRay bus. In this case, the CC updates the slot status field of the message buffer and triggers the status updated transition SU. With the SU transition, the CC sets the message buffer interrupt flag FR_MBCCSRn[MBIF] to indicate the successful message transmission.

Depending on the Transmission Mode flag FR_MBCCFRn[MTM], the CC changes the commit flag FR_MBCCSRn[CMT] and the valid flag FR_MBCCSRn[DVAL]. If the FR_MBCCFRn[MTM] flag is negated, the message buffer is in *Event Transmission mode*. In this case, each committed message is transmitted only once. The commit flag FR_MBCCSRn[CMT] is cleared with the SU transition. If the FR_MBCCFRn[MTM] flag is asserted, the message buffer is in the *State Transmission mode*. In this case, each committed message is transmitted as long as the application provides new data or locks the message buffers. The CC will not clear the FR_MBCCSRn[CMT] flag at the end of transmission and will set the valid flag FR_MBCCSRn[DVAL] to indicate that the message will be transmitted again.

51.6.6.2.7.2 Message buffer status update after incomplete message transmission

The term *incomplete message transmission* refers to the fact that not all payload data that should be transmitted was sent to FlexRay bus. This may be caused by the following regular conditions in the dynamic segment:

1. The transmission slot starts in a minislot with a minislot number greater than [pLatestTx](#).
2. The transmission slot did not exist in the dynamic segment at all.

Additionally, an incomplete message transmission can be caused by internal communication errors. If those error occur, the Protocol Engine Communication Failure Interrupt Flag PECF_IF is set in the [Protocol Interrupt Flag Register 1 \(FR_PIFR1\)](#).

In either of these two cases, the status of the message buffer is not changed at all with the SU transition. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

51.6.6.2.7.3 Message buffer status update after null frame transmission

After the transmission of a null frame, the status of the message buffer that was used for the null frame transmission is not changed at all. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

51.6.6.3 Receive message buffers

The section provides a detailed description of the functionality of the receive message buffers. If receive message buffers are used, the related receive shadow buffer must be configured as described in [Section 51.6.3.2, "Receive shadow buffers"](#).

A receive message buffer is used to receive a message from the FlexRay bus based on individual filter criteria. The CC uses the receive message buffer to provide the following data to the application:

- Message data received
- Information about the reception process
- Status information about the slot in which the message was received

A individual message buffer with message buffer number n is configured as a receive message buffer by the following configuration settings:

- $FR_MBCCSRn[MTD] = 0$ (receive message buffer)

Both the application and the CC have access to certain message buffers. To ensure data consistency, a message buffer locking scheme controls the access to the data, control, and status bits of a message buffer. The access regions for receive message buffers are depicted in [Figure 1081](#). A description of the regions is given in [Table 1084](#). If an region is active as indicated in [Table 1085](#), the access scheme given for that region applies to the message buffer.

Figure 1081. Receive message buffer access regions

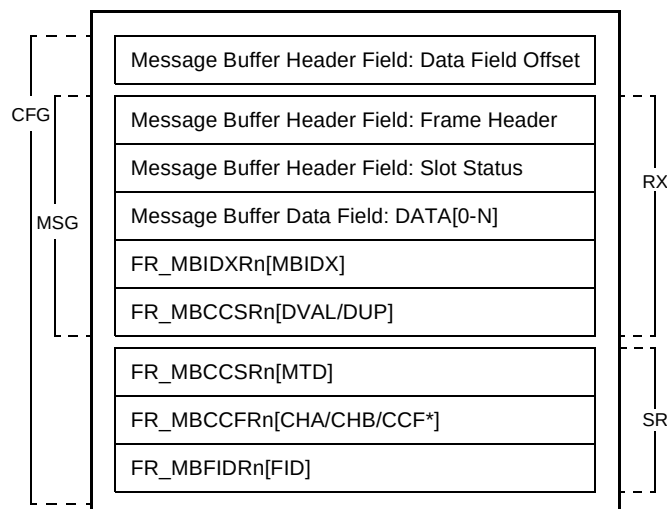


Table 1084. Receive message buffer access region description

Region	Access from		Region used for
	Application	Module	
CFG	Read/write	—	Message buffer configuration, message data, and status access
MSG	Read/write	—	Message data, header, and status access
RX	—	Write-only	Message reception and status update
SR	—	Read-only	Message buffer search data

The trigger bits $FR_MBCCSRn[EDT]$ and $FR_MBCCSRn[LCKT]$ and the interrupt enable bit $FR_MBCCSRn[MBIE]$ are not under access control and can be accessed from the application at any time. The status bits $FR_MBCCSRn[EDS]$ and $FR_MBCCSRn[LCKS]$ are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC set access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The receive message buffer states are given in [Figure 1082](#). A description of the message buffer states is given in [Table 1080](#), which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required status information. The internal status information is not visible to the application.

Figure 1082. Receive message buffer states

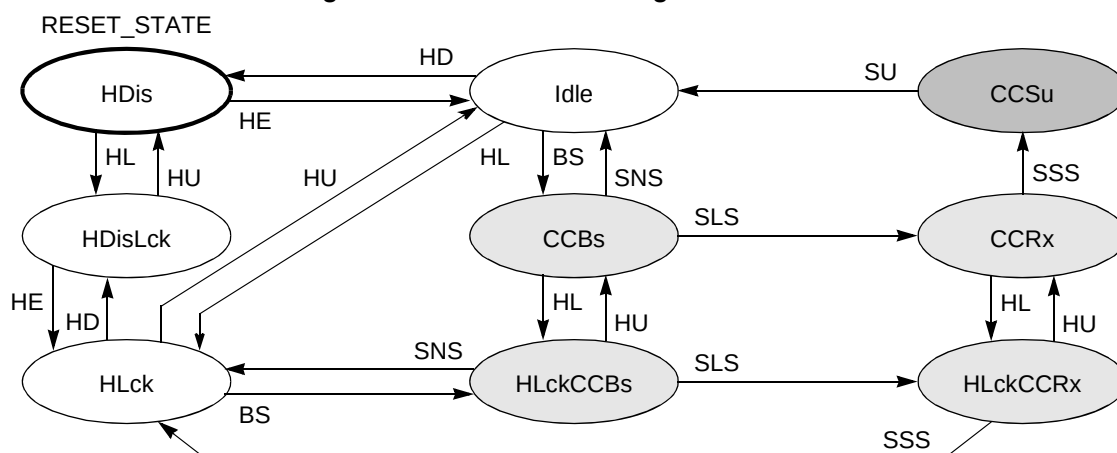


Table 1085. Receive message buffer states and access

State	FR_MBCCSRn		Access from		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	—	SR	Idle —Message buffer is idle. Included in message buffer search.
HDis	0	0	CFG	—	Disabled —Message buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	—	Disabled and Locked —Message buffer under configuration. Excluded from message buffer search.
HLck	1	1	MSG	—	Locked —Application access to data, control, and status. Included in message buffer search.
CCBs	1	0	—	—	Buffer Subscribed —Message buffer subscribed for reception. Filter matches next (slot, cycle, channel) tuple.
HLckCCBs	1	1	MSG	—	Locked and Buffer Subscribed —Application access to data, control, and status. Message buffer subscribed for reception.
CCRx	1	0	—	—	Message Receive —Message data received into related shadow buffer.

Table 1085. Receive message buffer states and access (continued)

State	FR_MBCCSRn		Access from		Description
	EDS	LCKS	Appl.	Module	
HLckCCR _x	1	1	MSG	—	Locked and Message Receive—Application access to data, control, and status. Message data received into related shadow buffer.
CCSu	1	0	—	RX	Status Update —Message buffer status update. Update of status flags, the slot status field, and the header index.

51.6.6.3.1 Message buffer transitions

51.6.6.3.1.1 Application transitions

The application transitions that can be triggered by the application using the commands are described in [Table 1081](#). The application issues the commands by writing to the [Message Buffer Configuration, Control, Status Registers \(FR_MBCCSRn\)](#). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

51.6.6.3.1.2 Message buffer enable and disable

The enable and disable commands are issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in the CCBs, HLckCCBs, or CCR_x state, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

51.6.6.3.1.3 Message buffer lock and unlock

The lock and unlock commands are issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL while the message buffer is in the CCR_x state, the lock transition has no effect (command is ignored) and the message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the [CHI Error Flag Register \(FR_CHIERFR\)](#) is set.

Table 1086. Receive message buffer application transitions

Transition	Host command	Condition	Description
HE	FR_MBCCSRn[EDT] := 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable
HL	FR_MBCCSRn[LCKT] := 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock

51.6.6.3.1.4 Module Transitions

The module transitions that can be triggered by the CC are described in [Table 1087](#). Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 1087. Receive message buffer module transitions

Transition	Condition	Description
BS	Slot match and cyclecounter match	Buffer Subscribed—The message buffer filter matches next slot and cycle.
SLS	Slot start	Slot Start—Start of either Static Slot or Dynamic Slot.
SNS	Symbol window start or NIT start	Symbol Window or NIT Start—Start of either Symbol Window or NIT.
SSS	Slot start, symbol window start, or NIT start	Slot or Segment Start—Start of either Static Slot, Dynamic Slot, Symbol Window, or NIT.
SU	Status updated	Status Updated—Slot Status field, message buffer status flags, header index updated. Interrupt flag set.

51.6.6.3.1.5 Transition priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in [Table 1088](#), the module transitions have a higher priority than the application transitions. For all states except the CCRx state, a module transition and the application lock/unlock transition HL/HU can be executed at the same time. The result state is reached by first applying the module transition and subsequently the application transition to the intermediately reached state. For example, if the message buffer is in the CCBs buffer subscribed state and the module triggers the SLS slot start transition at the same time that the application locks the message buffer through the HL transition, the intermediate state is CCRx and the resulting state is locked buffer subscribed state, HLckCCRx.

Table 1088. Receive message buffer transition priorities

State	Priorities	Description
Module vs. Application		
Idle	BS > HD	Buffer subscribed > Message buffer disable
HLck	BS > HD	Buffer subscribed > Message buffer disable
CCRx	SSS > HL	Slot or segment start > Message buffer lock

51.6.6.3.2 Message reception

As a result of the message buffer search, the CC changes the state of up to two enabled receive message buffers from either idle state Idle or locked state HLck to either subscribed state CCBs or locked buffer subscribed state HLckCCBs by triggering the buffer subscribed transition BS.

If the receive message buffers for the next slot are assigned to both channels, then at most one receive message buffer is changed to a buffer subscribed state.

If more than one matching message buffer is assigned to a certain channel, then only the message buffer with the lowest message buffer number is in one of the states mentioned above.

With the start of the next static or dynamic slot, the module triggers the slot start transition SLS. This changes the state of the subscribed receive message buffers from either CCBs to CCRx or from HLckCCBs to HLckCCRx, respectively.

During the reception slot, the received frame data is written into the shadow buffers. For details on receive shadow buffers, see [Section 51.6.6.3.5, "Receive shadow buffer concept"](#). The data and status of the receive message buffers that are the CCRx or HLckCCRx are not modified in the reception slot.

51.6.6.3.3 Message buffer update

With the start of the next static or dynamic slot or with the start of the symbol window or NIT, the module triggers the slot or segment start transition SSS. This transition changes the state of the receiving receive message buffers from either CCRx to CCSu or from HLckCCRx to HLck, respectively.

If a message buffer is in the locked state HLckCCRx, no update will be performed. The received data is lost. This is indicated by setting the Frame Lost Channel A/B Error Flag FRLA_EF/FRLB_EF in the [CHI Error Flag Register \(FR_CHIERFR\)](#).

If a message buffer was in the CCRx state, it is now in the CCSu state. After the evaluation of the slot status provided by the PE, the message buffer is updated. The message buffer update depends on the slot status bits and the segment the message buffer is assigned to, as described in [Table 1091](#).

Table 1089. Receive message buffer update

vSS!ValidFrame	vRF!Header!NFIndicator	Update description
1	1	Valid non-null frame received. – Message Buffer Data field updated. – Frame Header field updated. – Slot Status field updated. – DUP = 1 – DVAL = 1 – MBIF = 1
1	0	Valid null frame received. – Message Buffer Data field <i>not</i> updated. – Frame Header field <i>not</i> updated. – Slot Status field updated. – DUP = 0 – DVAL <i>not</i> changed – MBIF = 1

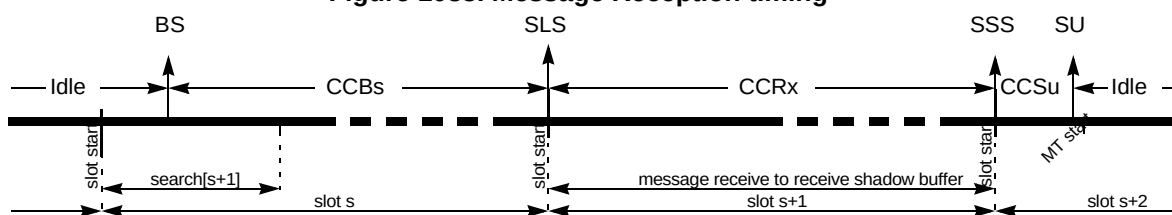
Table 1089. Receive message buffer update (continued)

vSS!ValidFrame	vRF!Header!NFIndicator	Update description
0	x	No valid frame received. – Message Buffer Data field not updated. – Frame Header field not updated. – Slot Status field updated. – DUP = 0 – DVAL <i>not</i> changed. – MBIF = 1, if the slot was not an empty dynamic slot. <i>Note: An empty dynamic slot is indicated by the following frame and slot status bit values:</i> vSS!ValidFrame = 0 and vSS!SyntaxError = 0 and vSS!ContentError = 0 and vSS!BViolation = 0.

Note: If the number of the last slot in the current communication cycle on a given channel is n , then all receive message buffers assigned to this channel with $FR_MBFIDRn[FID] > n$ will not be updated at all.

When the receive message buffer update has finished the status updated transition SU is triggered, which changes the buffer state from CCSu to Idle. An example receive message buffer timing and state change diagram for a normal frame reception is given in [Figure 1083](#).

Figure 1083. Message Reception timing



The amount of message data written into the message buffer data field of the receive shadow buffer is determined by the following items:

- The message buffer segment that the message buffer is assigned to, as defined by the [Message Buffer Segment Size and Utilization Register \(FR_MBSSUTR\)](#)
- The message buffer data field size, as defined by the related field of the [Message Buffer Data Size Register \(FR_MBDSR\)](#)
- The number of bytes received over the FlexRay bus

If the message buffer is assigned to the message buffer segment 1, and the number of received bytes is greater than $2 \times FR_MBDSR.MBSEG1DS$, the CC writes only $2 \times FR_MBDSR.MBSEG1DS$ bytes into the message buffer data field of the receive shadow buffer. If the number of received bytes is less than $2 \times FR_MBDSR.MBSEG1DS$, the CC writes only the received number of bytes and will not change the trailing bytes in the message buffer data field of the receive shadow buffer. The same holds for the message buffer segment 2 with $FR_MBDSR.MBSEG2DS$.

51.6.6.3.4 Received message access

To access the message data received over the FlexRay bus, the application reads the message data stored in the message buffer data field of the corresponding receive message buffer. The access to the message buffer data field is described in [Section 51.6.3.1, "Individual Message Buffers"](#).

The application can read the message buffer data field if the receive message buffer is one of the states HDis, HDisLck, or HLck. If the message buffer is in one of these states, the CC will not change the content of the message buffer.

51.6.6.3.5 Receive shadow buffer concept

The receive shadow buffer concept applies only to individual receive message buffers. The intention of this concept is to ensure that only syntactically and semantically valid received non-null frames are presented to the application in a receive message buffer. The basic structure of a receive shadow buffer is described in [Section 51.6.3.2, "Receive shadow buffers"](#).

The receive shadow buffers temporarily store the received frame header and message data. The slot status information is generated after the slot boundary. If the slot status information indicates the reception of the valid non-null frame (see [Table 1089](#)), the CC writes the slot status into the slot status field of the receive shadow buffer and exchanges the content of the [Message Buffer Index Registers \(FR_MBIDX_{Rn}\)](#) with the content of the corresponding internal shadow buffer index register. In all other cases, the CC writes the slot status into the identified receive message buffer, depending on the slot status and the FlexRay segment the message buffer is assigned to.

The shadow buffer concept, with its index exchange, results in the fact that the FlexRay memory area located message buffer associated to an individual receive message buffer changes after successful reception of a valid frame. This means that the message buffer area in the FlexRay memory area accessed by the application for reading the received message is different from the initial setting of the message buffer. Therefore, the application must not rely on the index information written initially into the [Message Buffer Index Registers \(FR_MBIDX_{Rn}\)](#). Instead, the index of the message buffer header field must be fetched from the [Message Buffer Index Registers \(FR_MBIDX_{Rn}\)](#).

51.6.7 Individual message buffer search

This section provides a detailed description of the message buffer search algorithm.

The message buffer search determines for each enabled channel whether a slot s in a communication cycle c is assigned for frame or null frame transmission or if it is subscribed for frame reception on that channel.

The message buffer search is a sequential algorithm invoked at the following protocol-related events:

- NIT start
- Slot start in the static segment
- Minislot start in the dynamic segment

The message buffer search within the NIT searches for message buffers assigned or subscribed to slot 1. The message buffer search within slot n searches for message buffers assigned or subscribed to slot $n + 1$.

In general, the message buffer search for the next slot n considers only message buffers that

1. Are enabled (that is, $\text{FR_MBCCSRn[EDS]} = 1$), and
2. Match the next slot n , (that is, $\text{FR_MBFIDRn[FID]} = n$)

On top of that, for the static segment, only those message buffers are considered that match the condition of at least one row of [Table 1090](#). For the dynamic segment, only those message buffers are considered that match the condition of at least one row of [Table 1091](#). These message buffers are called *matching* message buffers.

For each enabled channel, the message buffer search may identify multiple *matching* message buffers. Among all matching message buffers, the message buffers with highest priority according to [Table 1090](#) for the static segment and according to [Table 1091](#) for the dynamic segment are selected.

Table 1090. Message buffer search priority (static segment)

Priority	MTD	LCKS	CMT	CCFM (1)	Description	Transition
(highest) 0	1	0	1	1	Transmit buffer, matches cycle count, not locked and committed	MA
1	1	—	0	1	Transmit buffer, matches cycle count, not committed	SA
	1	1	—	1	Transmit buffer, matches cycle count, locked	SA
2	1	—	—	—	Transmit buffer	SA
3	0	0	n/a	1	Receive buffer, matches cycle count, not locked	SB
(lowest) 4	0	1	n/a	1	Receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see [Section 51.6.7.1, "Message buffer cycle counter filtering"](#).

Table 1091. Message buffer search priority (dynamic segment)

Priority	MTD	LCKS	CMT	CCFM (1)	Description	Transition
(highest) 0	1	0	1	1	Transmit buffer, matches cycle count, not locked and committed	MA
1	0	0	n/a	1	Receive buffer, matches cycle count, not locked	SB
(lowest) 2	0	1	n/a	1	Receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see [Section 51.6.7.1, "Message buffer cycle counter filtering"](#)

If there are multiple message buffer with highest priority, the message buffer with the lowest message buffer number is selected. All message buffers that have the highest priority must have a consistent channel assignment as specified in [Section 51.6.7.2, "Message buffer channel assignment consistency"](#).

Depending on the message buffer channel assignment, the same message buffer can be found for both channel A and channel B. In this case, this message buffer is used as described in [Section 51.6.3.1, "Individual Message Buffers"](#).

51.6.7.1 Message buffer cycle counter filtering

The message buffer cycle counter filter is a value-mask filter defined by the CCFE, CCFMSK, and CCFVAL fields in the [Message Buffer Cycle Counter Filter Registers \(FR_MBCCFRn\)](#). This filter determines a set of communication cycles in which the message buffer is considered for message reception or message transmission. If the cycle counter filter is disabled (CCFE = 0), this set of cycles consists of all communication cycles.

If the cycle counter filter of a message buffer does not match a certain communication cycle number, this message buffer is not considered for message transmission or reception in that communication cycle. In case of a transmit message buffer assigned to a slot in the static segment, though, this buffer is added to the matching message buffers to indicate the slot assignment and to trigger the null frame transmission.

The cycle counter filter of a message buffer matches the communication cycle with the number CYCCNT if at least one of the following conditions evaluates to true:

Equation 50 $MBCCFRn[CCFE] = 0$

Equation 51

$$CYCCNT \& MBCCFRn[CCFMSK] = MBCCFRn[CCFVAL] \& MBCCFRn[CCFMSK]$$

51.6.7.2 Message buffer channel assignment consistency

The message buffer channel assignment given by the CHA and CHB bits in the [Message Buffer Cycle Counter Filter Registers \(FR_MBCCFRn\)](#) defines the channels on which the message buffer will receive or transmit. The message buffer with number n transmits or receives on channel A if $FR_MBCCFRn[CHA] = 1$ and transmits or receives on channel B if $FR_MBCCFRn[CHB] = 1$.

To ensure correct message buffer operation, all message buffers assigned to the same slot and with the same priority must have a *consistent* channel assignment. That means they must be either assigned to one channel only, or must be assigned to *both* channels. The behavior of the message buffer search is not defined if both types of channel assignments occur for one slot and priority. An inconsistent channel assignment for message buffer 0 and message buffer 1 is depicted in [Figure 1084](#).

Figure 1084. Inconsistent channel assignment

MB0	FR_MBFIDR0[FID] = 10	FR_MBCCFR0[CHA] = 1, FR_MBCCFR0[CHB] = 0	 single channel assignment
MB1	FR_MBFIDR1[FID] = 10	FR_MBCCFR1[CHA] = 1, FR_MBCCFR1[CHB] = 1	

51.6.7.3 Node-related slot multiplexing

The term *node-related slot multiplexing* applies to the dynamic segment only and refers to the functionality if transmit as well as receive message buffers are configured for the same slot.

According to [Table 1091](#) the transmit buffer is only found if the cycle counter filter matches, and the buffer is not locked and committed. In all other cases, the receive buffer will be found. Thus, if the block has no data to transmit in a dynamic slot, it is able to receive frames on that slot.

51.6.7.4 Message buffer search error

There are two kinds of errors that may occur during message buffer search^(r).

51.6.7.4.1 Message buffer search start while running

If the message buffer search is running in slot $n - 1$ and the next message buffer search start event appears due to the start of slot n , the message buffer search engine is stopped and the Message Buffer Search Error Flag MBS_EF is set in the [CHI Error Flag Register \(FR_CHIERFR\)](#). As a result of this stop, no individual message buffer is identified for transmission or reception in slot n . Additionally, the search engine will not be started in slot n , and consequently no individual message buffer is identified for transmission or reception in slot $n + 1$.

A message buffer search error appears only if the CHI frequency is too slow to allow the search through all message buffers to be completed within the NIT or a minislot.

For more details of minimum required CHI frequency see [Section 51.7.5, "Number of usable message buffers"](#).

51.6.7.4.2 Illegal message buffer index found

If the message buffer search has finished the message buffer search in slot $n - 1$, it retrieves the data offset values for the found message buffers and the receive shadow buffers. If one of these message buffers contains an illegal message buffer index, the Message Buffer Search Error Flag MBS_EF is set in the [CHI Error Flag Register \(FR_CHIERFR\)](#) and no individual message buffer is identified for transmission or reception in slot n . The legal message buffer index values for the individual and receive shadow buffers are specified in [Section 51.5.2.52, "Receive Shadow Buffer Index Register \(FR_RSBR\)"](#) and [Section 51.5.2.82, "Message Buffer Index Registers \(FR_MBIDXRn\)"](#).

51.6.8 Individual message buffer reconfiguration

The initial configuration of each individual message buffer can be changed even when the protocol is not in the *POC:config* state. This is referred to as individual message buffer *reconfiguration*. The configuration bits and fields that can be changed are given in the section on [Specific Configuration Data](#). The common configuration data given in the section on [Specific Configuration Data](#) can not be reconfigured when the protocol is out of the *POC:config* state.

51.6.8.1 Reconfiguration schemes

Depending on the target and destination basic state of the message buffer that is to be reconfigured, there are three reconfiguration schemes.

51.6.8.1.1 Basic type not changed (RC1)

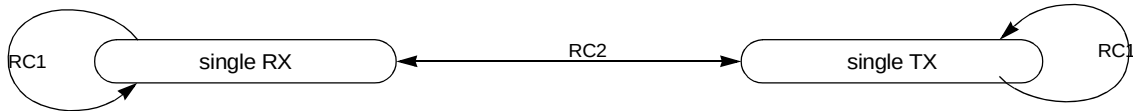
A reconfiguration will not change the basic type of the individual message buffer, if the message buffer transfer direction bit FR_MBCCSRn[MTD] is not changed. This type of reconfiguration is denoted by RC1 in [Figure 1085](#). Transmit and receive message buffers can be RC1-reconfigured when in the HDis or HDisLck state.

r. The FIFO reception is not affected by the search errors. Additionally, if no rx buffer has been found due to an search error, the received frame is considered for FIFO reception.

51.6.8.1.2 Buffer type not changed (RC2)

A reconfiguration will not change the buffer type of the individual message buffer. This type of reconfiguration is denoted by RC2 in [Figure 1085](#). It applies to transmit and receive message buffers. Transmit and receive message buffers can be RC2-reconfigured when in the HDis or HDisLck state.

Figure 1085. Message buffer reconfiguration scheme



51.6.9 Receive FIFOs

This section provides the functional description of the two receive FIFOs.

51.6.9.1 Overview

The two receive FIFOs implement the queued message buffer concept defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. One FIFO is assigned to channel A, and the other FIFO is assigned to channel B. Both FIFOs work independently from each other.

The message buffer structure of each FIFO is described in [Section 51.6.3.3, "Receive FIFO"](#). The area in the FlexRay memory area for each of the two FIFOs is characterized by:

- The FIFO system memory base address
- The index of the first FIFO entry given by the [Receive FIFO Start Index Register \(FR_RFSIR\)](#)
- The data field offset of the data field belonging to the first FIFO entry given by the [Receive FIFO Start Data Offset Register \(FR_RFSDOR\)](#)
- The number of FIFO entries and the length of each FIFO entry as given by the [Receive FIFO Depth and Size Register \(FR_RFDSR\)](#)

51.6.9.2 FIFO configuration

The FIFOs can be configured for two different locations of the system memory base address via the FIFO Address Mode bit FAM in the [Module Configuration Register \(FR_MCR\)](#).

51.6.9.2.1 Single System Memory Base Address mode

This mode is configured when the FIFO address mode flag FR_MCR[FAM] is set to 0. In this mode, the location of the system memory base address for the FIFO buffers is the [System Memory Base Address Register \(FR_SYMBADR\)](#).

51.6.9.2.2 Dual System Memory Base Address mode

This mode is configured when the FIFO address mode flag FR_MCR[FAM] is set to 1. In this mode, the location of the system memory base address for the FIFO buffers is the [Receive FIFO System Memory Base Address Register \(FR_RFSYMBADR\)](#).

The FIFO control and configuration data is given in [Section 51.6.3.7, "Receive FIFO control and configuration data"](#). The configuration of the FIFOs consists of two steps.

The first step is the allocation of the required amount of FlexRay memory area for the FlexRay window. This includes the allocation of the message buffer header area and the allocation of the message buffer data fields. For more details, see [Section 51.6.4, “FlexRay memory area layout”](#).

The second step is the programming of the configuration data register while the PE is in *POC:config*.

The following steps configure the layout of the FIFO.

1. Configure the FIFO update and address modes in [Module Configuration Register \(FR_MCR\)](#).
2. Configure the FIFO system memory base address.
3. Configure the [Receive FIFO Start Index Register \(FR_RFSIR\)](#) with the first message buffer header index that belongs to the FIFO.
4. Configure the [Receive FIFO Start Data Offset Register \(FR_RFSDOR\)](#) with the data field offset of the data field belonging to the first message buffer that belongs to the FIFO.
5. Configure the [Receive FIFO Depth and Size Register \(FR_RFDSR\)](#) with FIFO entry size.
6. Configure the [Receive FIFO Depth and Size Register \(FR_RFDSR\)](#) with FIFO depth.
7. Configure the FIFO filters.

51.6.9.3 FIFO periodic timer

The FIFO periodic timer is used to generate a FIFO almost-full interrupt at certain point in time, if the almost-full watermark is not reached but the FIFO is not empty. This can be used to prevent frames from getting stuck in the FIFO for a long time.

The FIFO periodic timer is configured via the [Receive FIFO Periodic Timer Register \(FR_RFPTR\)](#). If the periodic timer duration `FR_RFPTR[PTD]` is configured to `0x0000`, the periodic timer is continuously expired. If the periodic timer duration `FR_RFPTR[PTD]` is configured to `0x3FFF`, the periodic timer never expires. If the periodic timer is configured to a value *ptd*, greater than `0x0000` and smaller `0x3FFF`, the periodic timer expires and is restarted at the start of every communication cycle, and expires and is restarted after *ptd* macroticks have been elapsed.

51.6.9.4 FIFO reception

The FIFO reception is a CC internal operation.

A message frame reception is directed into the FIFO if no individual message buffer is assigned for transmission or subscribed for reception for the current slot. In this case, the FIFO filter path shown in [Figure 1086](#) is activated.

If the FIFO filter path indicates that the received frame has to be appended to the FIFO and the FIFO is not full, the CC writes the received frame header into the message buffer header field indicated by the CC internal FIFO write index. The frame payload data are written into the corresponding message buffer data field. If the status of the received frame indicates a valid non-null frame, the slot status information is written into the message buffer header field and the CC internal FIFO write index is updated by 1 and the fifo fill level FLA (FLB) in the [Receive FIFO Fill Level and POP Count Register \(FR_RFFLPCR\)](#) is incremented. If the status of the received frame indicates an invalid or null frame, the frame is not appended to the FIFO.

51.6.9.5 FIFO almost-full interrupt generation

If the FIFO fill level FLA (FLB) is updated after a frame reception and exceeds the FIFO watermark level WM (that is, $FLA > WM_A$ ($FLB > WM_B$)), then the FIFO almost-full interrupt flag $FR_GIFER[FAFAIF]$ ($FR_GIFER[FAFBIF]$) is asserted. If the periodic timer expires, and FIFOA (FIFOB) is not empty (that is, $FLA > 0$ ($FLB > 0$)), then the FIFO almost-full interrupt flag $FR_GIFER[FAFAIF]$ ($FR_GIFER[FAFBIF]$) is asserted.

51.6.9.6 FIFO overflow error generation

If the FIFOA (FIFOB) is full (that is, $FLA = FIFO_DEPTH_A$ ($FLB = FIFO_DEPTH_B$)) and the conditions for a FIFO reception as described in [Section 51.6.9.4, "FIFO reception"](#) are fulfilled, then the FIFO overflow error flag $FR_CHIERFR[FOVA_EF]$ ($FR_CHIERFR[FOVB_EF]$) is asserted.

51.6.9.7 FIFO message access

The FIFOA (FIFOB) contains valid messages if the FIFO fill level given in the fields FLA (FLB) in the [Receive FIFO Fill Level and POP Count Register \(\$FR_RFFLPCR\$ \)](#) is greater than 0. The [Receive FIFO A Read Index Register \(\$FR_RFARIR\$ \)](#) and the [Receive FIFO B Read Index Register \(\$FR_RFBIRIR\$ \)](#) point to a message buffer with valid content and the oldest frames stored in the FIFO. The respective read data field offsets can be calculated according to [Equation 1-125](#).

If the FIFO fill level FLA (FLB) in the [Receive FIFO Fill Level and POP Count Register \(\$FR_RFFLPCR\$ \)](#) is 0, then the FIFOA (FIFOB) contains no valid messages and the corresponding read index register [Receive FIFO A Read Index Register \(\$FR_RFARIR\$ \)](#) or [Receive FIFO B Read Index Register \(\$FR_RFBIRIR\$ \)](#) point to a message buffer with invalid content. In this case, the application must not read data from this FIFO.

To access the oldest message in the FIFOA (FIFOB), the application first reads the read index RDIDX out of the [Receive FIFO A Read Index Register \(\$FR_RFARIR\$ \)](#) ([Receive FIFO B Read Index Register \(\$FR_RFBIRIR\$ \)](#)). This read index points to the message buffer header field of the oldest message buffer that contains valid received message data. The data field offset belonging to this message buffer must be calculated by the application according to [Equation 1-125](#). The application can access the message data as described in [Section 51.6.3.3, "Receive FIFO"](#). When the application has read the message buffer data and status information, it can update the FIFO as described in [Section 51.6.9.8, "FIFO update"](#).

51.6.9.8 FIFO update

The application updates the FIFOA (FIFOB) by writing a pop count value pc different from 0 to the PCA (PCB) field in the [Receive FIFO Fill Level and POP Count Register \(\$FR_RFFLPCR\$ \)](#).

As a result of this operation, the CC removes the oldest pc entries from FIFOA (FIFOB).

If the specified pop count value pc is greater than the current fill level fl provided in FLA (FAB) field, then only fl entries are removed from the FIFOA (FIFOB), the remaining $fl - pc$ requested pop operations are discarded without any notification. In this case FIFOA (FIFOB) is empty after the update operation.

The read index in the [Receive FIFO A Read Index Register \(\$FR_RFARIR\$ \)](#) ([Receive FIFO B Read Index Register \(\$FR_RFBIRIR\$ \)](#)) is incremented by the number of removed items. If the

read index reaches the top of the FIFO, it wraps around to the FIFO start index defined in [Receive FIFO Start Index Register \(FR_RFSIR\)](#) automatically.

51.6.9.8.1 FIFO interrupt flag update

The FIFO Interrupt Flag Update mode is configured when the FIFO update mode flag FR_MCR[FUM] is set to 0. In this mode, FIFOA (FIFOB) will be updated by one entry when the interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is written with 1 by the application.

If the FIFO is empty, the update request is ignored without any notification.

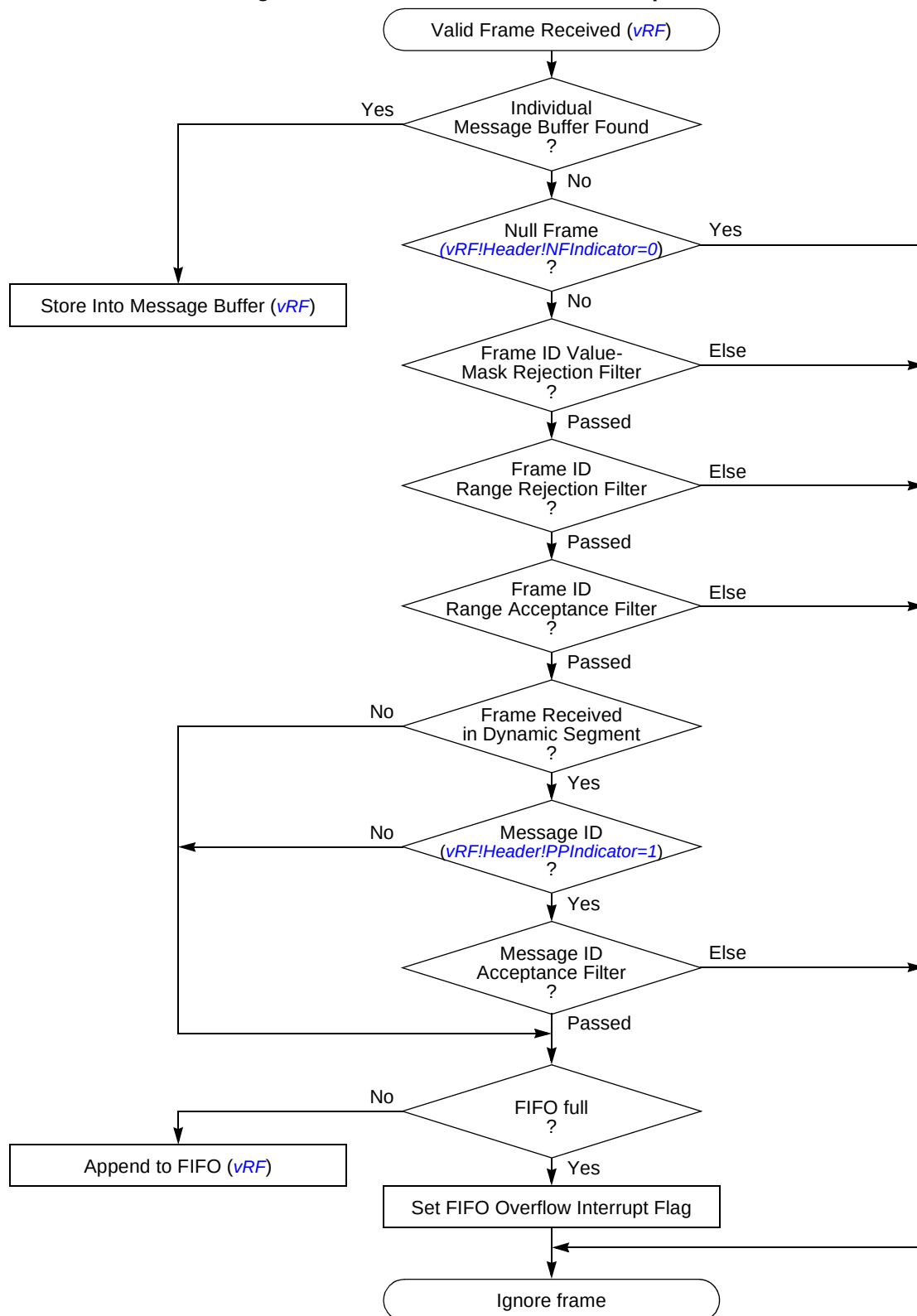
The read index in the [Receive FIFO A Read Index Register \(FR_RFARIR\)](#) ([Receive FIFO B Read Index Register \(FR_RFBIRIR\)](#)) is incremented by 1 if the FIFO was not empty. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index automatically.

51.6.9.9 FIFO filtering

The FIFO filtering is activated after all enabled individual receive message buffers have been searched without success for a message buffer to receive the current frame.

The CC provides three sets of FIFO filters. The FIFO filters are applied to valid non-null frames only. The FIFO will not receive invalid or null-frames. For each FIFO filter, the pass criteria is specified in the related section given below. Only frames that have passed all filters will be appended to the FIFO. The FIFO filter path is depicted in [Figure 1086](#).

Figure 1086. Received frame FIFO filter path



A received frame passes the FIFO filtering if it has passed all three type of filter.

51.6.9.9.1 RX FIFO Frame ID Value-mask Rejection filter

The Frame ID Value-mask Rejection filter is a value-mask filter and is defined by the fields in the [Receive FIFO Frame ID Rejection Filter Value Register \(FR_RFFIDRFVR\)](#) and the [Receive FIFO Frame ID Rejection Filter Mask Register \(FR_RFFIDRFMR\)](#). Each received frame with a frame ID FID that does not match the value-mask filter value passes the filter (that is, it is not rejected).

Consequently, a received valid frame with the frame ID FID passes the RX FIFO Frame ID Value-mask Rejection filter if [Equation 52](#) is fulfilled.

Equation 52

$$\begin{aligned} & \text{FID} \& \text{FR_RFFIDRFMR}[\text{FIDRFMSK}] \neq \\ & \neq \text{FR_RFFIDRFVR}[\text{FIDRFVAL}] \& \text{FR_RFFIDRFMR}[\text{FIDRFMSK}] \end{aligned}$$

The RX FIFO Frame ID Value-mask Rejection filter can be configured to pass all frames by using the following settings:

- $\text{FR_RFFIDRFVR}[\text{FIDRFVAL}] = 0x000$ and $\text{FR_RFFIDRFMR}[\text{FIDRFMSK}] = 0x7FF$

Using the settings above, only the frame with frame ID 0 will be rejected, which is an invalid frame. All other frames will pass.

The RX FIFO Frame ID Value-mask Rejection filter can be configured to reject all frames by using the following settings:

- $\text{FR_RFFIDRFMR}[\text{FIDRFMSK}] = 0x000$

Using the settings above, [Equation 52](#) can never be fulfilled ($0 \neq 0$) and thus all frames are rejected; no frame will pass. This is the reset value for the RX FIFO.

51.6.9.9.2 RX FIFO Frame ID Range Rejection filter

Each of the four RX FIFO Frame ID Range filters can be configured as a rejection filter. The filters are configured by the [Receive FIFO Range Filter Configuration Register \(FR_RFRFCFR\)](#) and controlled by the [Receive FIFO Range Filter Control Register \(FR_RFRFCTR\)](#). The RX FIFO frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range Rejection filters if either no rejection filter is enabled, or, if for all of the enabled RX FIFO Frame ID Range Rejection filters (that is, $\text{FR_RFRFCTR}[\text{FiMD}] = 1$ and $\text{FR_RFRFCTR}[\text{FiEN}] = 1$), [Equation 53](#) is fulfilled.

Equation 53

$$(\text{FID} < \text{FR_RFRFCFR}_{\text{SEL}}[\text{SID}_{\text{IBD}} = 0]) \text{ or } (\text{FR_RFRFCFR}_{\text{SEL}}[\text{SID}_{\text{IBD}} = 1] < \text{FID})$$

Consequently, all frames with a frame ID that fulfills [Equation 54](#) for at least one of the enabled rejection filters will be rejected and thus not pass.

Equation 54

$$\text{FR_RFRFCFR}_{\text{SEL}}[\text{SID}_{\text{IBD}} = 0] \leq \text{FID} \leq \text{FR_RFRFCFR}_{\text{SEL}}[\text{SID}_{\text{IBD}} = 1]$$

51.6.9.9.3 RX FIFO Frame ID Range Acceptance filter

Each of the four RX FIFO Frame ID Range filters can be configured as an acceptance filter. The filters are configured by the [Receive FIFO Range Filter Configuration Register](#)

(*FR_RFRFCFR*) and controlled by the *Receive FIFO Range Filter Control Register (FR_RFRFCTR)*. The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range acceptance filters if either no acceptance filter is enabled, or, if for at least one of the enabled RX FIFO Frame ID Range acceptance filters (that is, $FR_RFRFCTR[FIMD] = 0$ and $FR_RFRFCTR[FIEN] = 1$), *Equation 55* is fulfilled.

Equation 55

$$FR_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR_RFRFCFR_{SEL}[SID_{IBD=1}]$$

51.6.9.9.4 RX FIFO Message ID Acceptance filter

The RX FIFO Message ID Acceptance filter is a value-mask filter and is defined by the *Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)* and the *Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)*. This filter applies only to valid frames received in the dynamic segment with the payload preamble indicator bit PPI set to 1. All other frames will pass this filter.

A received valid frame in the dynamic segment with the payload preamble indicator bit PPI set to 1 and with the message ID MID (the first two bytes of the payload) will pass the RX FIFO Message ID Acceptance filter if *Equation 56* is fulfilled.

Equation 56

$$\begin{aligned} MID \& FR_RFMIDAFMR[MIDAFMSK] = \\ &= FR_RFMIDAFMR[MIDAFVAL] \& FR_RFMIDAFMR[MIDAFMSK] \end{aligned}$$

The RX FIFO Message ID Acceptance filter can be configured to accept all frames by setting

- $FR_RFMIDAFMR[MIDAFMSK] := 0x000$

Using the settings above, *Equation 56* is always fulfilled and all frames will pass.

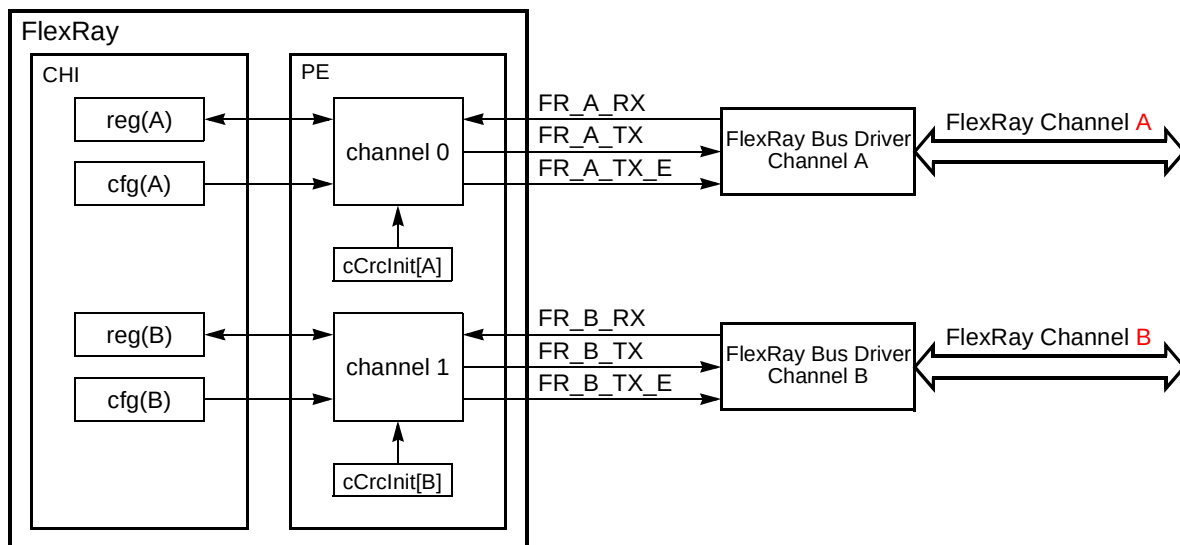
51.6.10 Channel device modes

This section describes the two FlexRay channel device modes that are supported by the CC.

51.6.10.1 Dual-Channel Device mode

In Dual-Channel Device mode, both FlexRay ports are connected to physical FlexRay bus lines. The FlexRay port consisting of FR_A_RX , FR_A_TX , and $FR_A_TX_EN$ is connected to the physical bus channel A and the FlexRay port consisting of FR_B_RX , FR_B_TX , and $FR_B_TX_EN$ is connected to the physical bus channel B. The dual-channel system is shown in *Figure 1087*.

Figure 1087. Dual-Channel Device mode



51.6.10.2 Single-Channel Device mode

Single-Channel Device mode supports devices that have only one FlexRay port available. This FlexRay port consists of the signals `FR_A_RX`, `FR_A_TX`, and `FR_A_TX_EN` and can be connected to either the physical bus channel A (shown in [Figure 1088](#)) or the physical bus channel B (shown in [Figure 1089](#)).

If the device is configured as a single-channel device by setting `FR_MCR[SCM]` to 1, only the internal channel A and FlexRay port A are used. Depending on the setting of `FR_MCR[CHA]` and `FR_MCR[CHB]`, the internal channel A behaves either as a FlexRay channel A or FlexRay channel B. The bit `FR_MCR[CHA]` must be set, if FlexRay port A is connected to FlexRay channel A. The bit `FR_MCR[CHB]` must be set if FlexRay port A is connected to FlexRay channel B. The two FlexRay channels differ only in the initial value for the frame CRC `cCrclnit`. For a single-channel device, the application can access and configure only the registers related to internal channel A.

Figure 1088. Single-Channel Device mode (channel A)

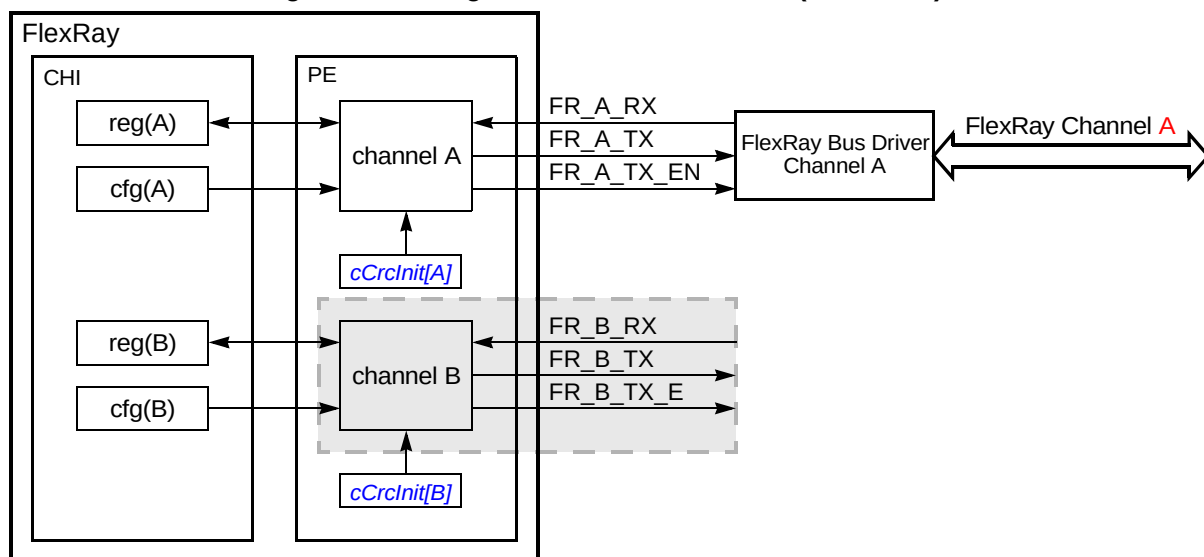
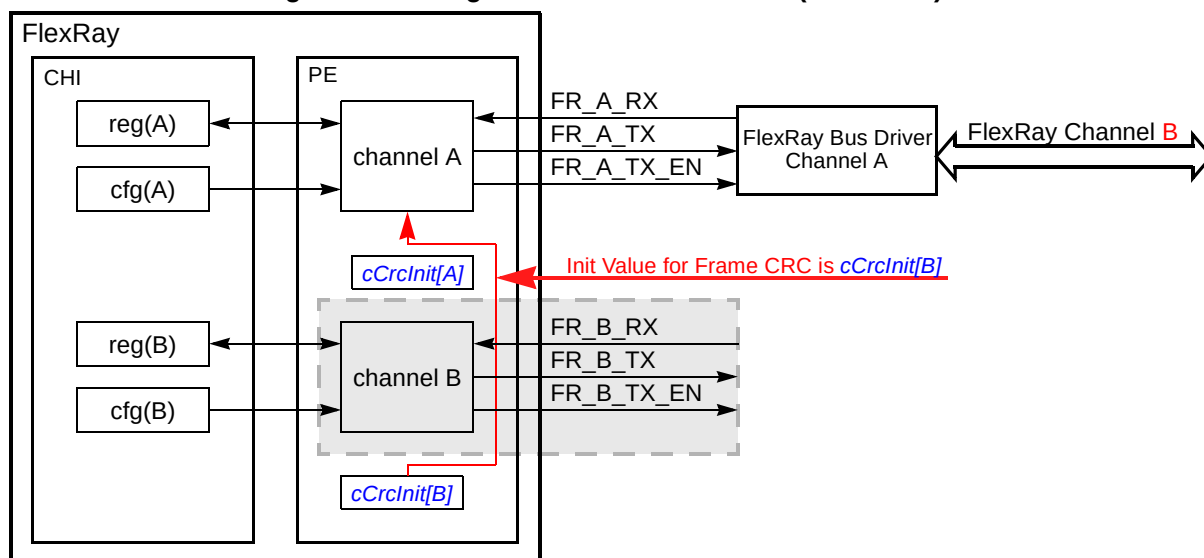


Figure 1089. Single-Channel Device mode (channel B)



51.6.11 External clock synchronization

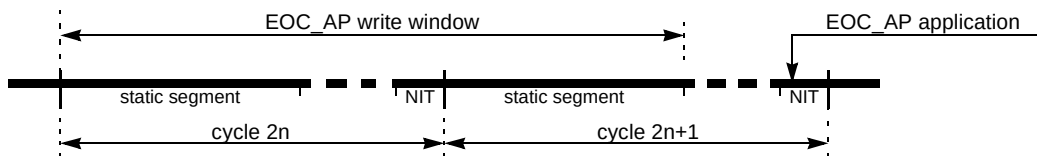
The application of the external rate and offset correction is triggered when the application writes to the EOC_AP and ERC_AP fields in the [Protocol Operation Control Register \(FR_POCR\)](#). The PE applies the external correction values in the next even-odd cycle pair as shown in [Figure 1090](#) and [Figure 1091](#).

Note: The values provided in the EOC_AP and ERC_AP fields are the values that were written from the application most recently. If these value were already applied, they will not be applied in the current cycle pair again.

If the offset correction applied in the NIT of cycle $2n + 1$ will be affected by the external offset correction, the EOC_AP field must be written to after the start of cycle $2n$ and before the end

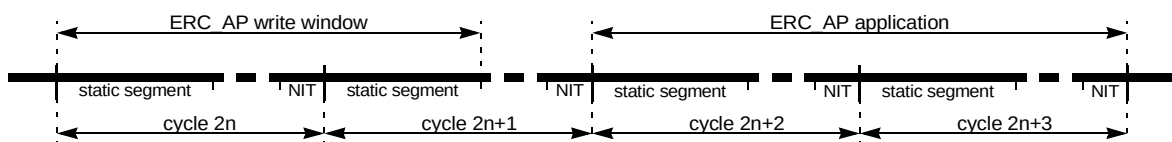
of the static segment of cycle $2n + 1$. If this field is written to after the end of the static segment of cycle $2n + 1$, it is not guaranteed that the external correction value is applied in cycle $2n + 1$. If the value is not applied in cycle $2n + 1$, then the value will be applied in the cycle $2n+3$. Refer to [Figure 1090](#) for timing details.

Figure 1090. External offset correction write and application timing



If the rate correction for the cycle pair $[2n + 2, 2n + 3]$ will be affected by the external offset correction, the ERC_AP field must be written to after the start of cycle $2n$ and before the end of the static segment of cycle $2n + 1$. If this field is written to after the end of the static segment of cycle $2n + 1$, it is not guaranteed that the external correction value is applied in cycle pair $[2n + 2, 2n + 3]$. If the value is not applied for cycle pair $[2n + 2, 2n + 3]$, then the value will be applied for cycle pair $[2n+4, 2n+5]$. Refer to [Figure 1091](#) for details.

Figure 1091. External rate correction write and application timing



51.6.12 Sync frame ID and sync frame deviation tables

The FlexRay protocol requires the provision of a snapshot of the synchronization frame ID tables for the even and odd communication cycle for both channels. The CC provides the means to write a copy of these internal tables into the FlexRay memory area and ensures application access to consistent tables by means of table locking. Once the application has locked the table successfully, the CC will not overwrite these tables and the application can read a consistent snapshot.

Note: Only synchronization frames that have passed the synchronization frame filters are considered for clock synchronization and appear in the sync frame tables.

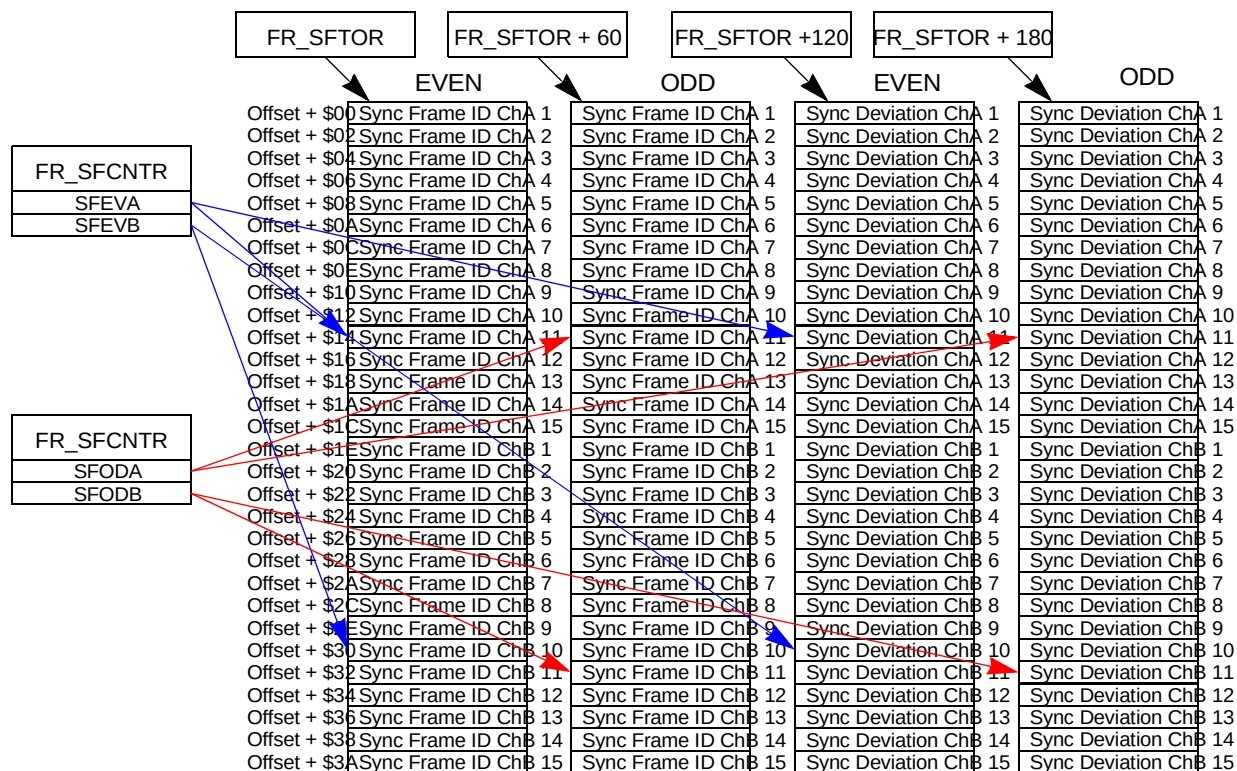
51.6.12.1 Sync frame ID table content

The sync frame ID table is a snapshot of the protocol-related variables [vsSyncIdListA](#) and [vsSyncIdListB](#) for each even and odd communication cycle. This table provides a list of the frame IDs of the synchronization frames received on the corresponding channel and cycle that are used for the clock synchronization.

51.6.12.2 Sync frame deviation table content

The sync frame deviation table is a snapshot of the protocol-related variable `zsDev(id)(oe)(ch)!Value`. Each sync frame deviation table entry provides the deviation value for the sync frame, with the frame ID presented in the corresponding entry in the sync frame ID table.

Figure 1092. Sync table memory layout



51.6.12.3 Sync frame ID and sync frame deviation table setup

The CC writes a copy of the internal synchronization frame ID and deviation tables into the FlexRay memory area if requested by the application. The application must provide the appropriate amount of FlexRay memory area for the tables. The memory layout of the tables is given in [Figure 1092](#). Each table occupies 120 16-bit entries.

While the protocol is in *POC:config* state, the application must program the offsets for the tables into the [Sync Frame Table Offset Register \(FR_SFTOR\)](#).

51.6.12.4 Sync frame ID and sync frame deviation table generation

The application controls the generation process of the sync frame ID and sync frame deviation tables into the FlexRay memory area using the [Sync Frame Table Configuration, Control, Status Register \(FR_SFTCCSR\)](#). A summary of the copy modes is given in [Table 1092](#).

Table 1092. Sync frame table generation modes

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	0	0	No sync frame table copy
0	0	1	Sync frame ID tables will be copied continuously
0	1	0	Reserved

Table 1092. Sync frame table generation modes (continued)

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	1	1	Sync frame ID tables and sync frame deviation tables will be copied continuously
1	0	0	No sync frame table copy
1	0	1	Sync frame ID tables for next even-odd-cycle pair will be copied
1	1	0	Reserved
1	1	1	Sync frame ID tables and sync frame deviation tables for next even-odd-cycle pair will be copied

The sync frame table generation process is described in the following for the even cycle. The same sequence applies to the odd cycle.

If the application has enabled the sync frame table generation by setting FR_SFTCCSR[SIDEN] to 1, the CC starts the update of the even-cycle-related tables after the start of the NIT of the next even cycle. The CC checks whether the application has locked the tables by reading the FR_SFTCCSR[ELKS] lock status bit. If this bit is set, the CC will not update the table in this cycle. If this bit is cleared, the CC locks this table and starts the table update. To indicate that these tables are currently updated and may contain inconsistent data, the CC clears the even table valid status bit FR_SFTCCSR[EVAL]. When all table entries related to the even cycle have been transferred into the FlexRay memory area, the CC sets the even table valid bit FR_SFTCCSR[EVAL] and the Even Cycle Table Written Interrupt Flag EVT_IF in the [Protocol Interrupt Flag Register 1 \(FR_PIFR1\)](#). If the interrupt enable flag EVT_IE is set, an interrupt request is generated.

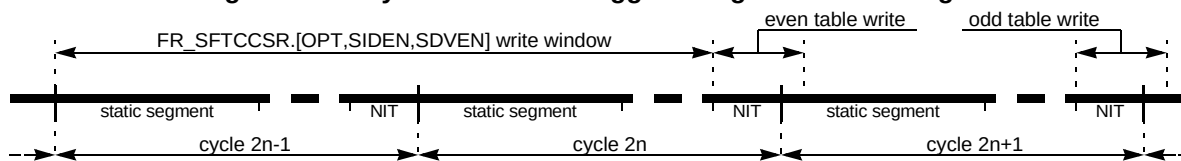
To read the generated tables, the application must lock the tables to prevent the CC from updating them. The locking is initiated by writing a 1 to the Even Table Lock trigger, FR_SFTCCSR[ELKT]. When the even table is not currently updated by the CC, the lock is granted and the even table lock status bit FR_SFTCCSR[ELKS] is set. This indicates that the application has successfully locked the even sync tables and the corresponding status information fields, SFRA and SFRB, in the [Sync Frame Counter Register \(FR_SFCNTR\)](#). The value in the FR_SFTCCSR[CYCNUM] field provides the number of the cycle that this table is related to.

The number of available table entries per channel is provided in the FR_SFCNTR[SFEVA] and FR_SFCNTR[SFEVB] fields. The application can now start to read the sync table data from the locations given in [Figure 1092](#).

After reading all the data from the locked tables, the application must unlock the table by writing to the Even Table Lock trigger FR_SFTCCSR[ELKT] again. The even table lock status bit FR_SFTCCSR[ELKS] is reset immediately.

If the sync frame table generation is disabled, the table valid bits FR_SFTCCSR[EVAL] and FR_SFTCCSR[EVAL] are reset when the counter values in the [Sync Frame Counter Register \(FR_SFCNTR\)](#) are updated. This is done because the tables stored in the FlexRay memory area are no longer related to the values in the [Sync Frame Counter Register \(FR_SFCNTR\)](#).

Figure 1093. Sync frame table trigger and generation timing



51.6.12.5 Sync frame table access

The sync frame tables will be transferred into the FlexRay memory area during the table write windows shown in [Figure 1092](#). During the table write, the application can not lock the table that is currently written. If the application locks the table outside of the table write window, the lock is granted immediately.

51.6.12.5.1 Sync Frame Table Locking and Unlocking

The application locks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT in the [Sync Frame Table Configuration, Control, Status Register \(FR_SFTCCSR\)](#). If the affected table is not currently written to the FlexRay memory area, the lock is granted immediately, and the lock status bit ELKS/OLKS is set. If the affected table is currently written to the FlexRay memory area, the lock is not granted. In this case, the application must issue the lock request again until the lock is granted.

The application unlocks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT. The lock status bit ELKS/OLKS is cleared immediately.

51.6.13 MTS generation

The CC provides a flexible means to request the transmission of the Media Access Test Symbol (MTS) in the symbol window on channel A or channel B.

The application can configure the set of communication cycles that transmit the MTS over the FlexRay bus by programming the CYCCNTMSK and CYCCNTVAL fields in the [MTS A Configuration Register \(FR_MTSACFR\)](#) and [MTS B Configuration Register \(FR_MTSBCFR\)](#).

The application enables or disables the generation of the MTS on either channel by setting or clearing the MTE control bit in the [MTS A Configuration Register \(FR_MTSACFR\)](#) or [MTS B Configuration Register \(FR_MTSBCFR\)](#). If an MTS is to be transmitted in a certain communication cycle, the application must set the MTE control bit during the static segment of the preceding communication cycle.

The MTS is transmitted over channel A in the communication cycle with number CYCCNT if [Equation 57](#), [Equation 58](#), and [Equation 59](#) are fulfilled.

$$\text{Equation 57} \quad \text{FR_PSR0[PROTSTATE]} = \text{POC:normal active}$$

$$\text{Equation 58} \quad \text{FR_MTSACRF[MTE]} = 1$$

$$\begin{aligned} \text{Equation 59} \quad & \text{CYCCNT} \& \text{FR_MTSACFR[CYCCNTMSK]} = \\ & = \text{FR_MTSACFR[CYCCNTVAL]} \& \text{FR_MTSACFR[CYCCNTMSK]} \end{aligned}$$

The MTS is transmitted over channel B in the communication cycle with number CYCCNT if [Equation 57](#), [Equation 60](#) and [Equation 61](#) are fulfilled.

Equation 60 $FR_MTSBCRF[MTE] = 1$

Equation 61 $CYCCNT \& FR_MTSBCFR[CYCCNTMSK] = FR_MTSBCFR[CYCCNTVAL] \& FR_MTSBCFR[CYCCNTMSK]$

51.6.14 Key slot transmission

51.6.14.1 Key slot assignment

A key slot is assigned to the CC if the key_slot_id field in the [Protocol Configuration Register 18 \(FR_PCR18\)](#) is configured with a value greater than 0 and less than or equal to number_of_static_slots in [Protocol Configuration Register 2 \(FR_PCR2\)](#); otherwise no key slot is assigned.

51.6.14.2 Key slot transmission in *POC:startup*

If a key slot is assigned and the CC is in the *POC:startup* state, startup null frames will be transmitted as specified by [FlexRay Communications System Protocol Specification, Version 2.1 Rev A](#).

51.6.14.3 Key slot transmission in *POC:normal active*

If a key slot is assigned and the CC is in *POC:normal active*, a frame of the type as shown in [Table 1093](#) is transmitted. If a transmit message buffer is configured for the key slot and a valid message is available, a message frame is transmitted (see [Section 51.6.6.2.5, "Message transmission"](#)). If no transmit message buffer is configured for the key slot or no valid message is available, a null frame is transmitted (see [Section 51.6.6.2.6, "Null frame transmission"](#)).

Table 1093. Key slot frame type

FR_PCR11[key_slot_used_for_sync]	FR_PCR11[key_slot_used_for_startup]	key slot frame type
0	0	Normal frame
0	1	Normal frame ⁽¹⁾
1	0	Sync frame
1	1	Startup frame

1. The frame transmitted has a semantically incorrect header and will be detected as an invalid frame at the receiver.

51.6.15 Sync frame filtering

Each received synchronization frame must pass the Sync Frame Acceptance filter and the Sync Frame Rejection filter before it is considered for clock synchronization. If the synchronization frame filtering is globally disabled (that is, the SFFE control bit in the [Module Configuration Register \(FR_MCR\)](#) is cleared), all received synchronization frames are considered for clock synchronization. If a received synchronization frame did not pass at least one of the two filters, this frame is processed as a normal frame and is not considered for clock synchronization.

51.6.15.1 Sync frame acceptance filtering

The synchronization frame acceptance filter is implemented as a value-mask filter. The value is configured in the [Sync Frame ID Acceptance Filter Value Register \(FR_SFIDAFVR\)](#) and the mask is configured in the [Sync Frame ID Acceptance Filter Mask Register \(FR_SFIDAFMR\)](#). A received synchronization frame with the frame ID FID passes the sync frame acceptance filter, if [Equation 62](#) or [Equation 63](#) evaluates to true.

Equation 62 $FR_MCR[SFFE] = 0$

Equation 63

$$FID \& FR_SFIDAFMR[FMSK] = \\ = FR_SFIDAFVR[FVAL] \& FR_SFIDAFMR[FMSK]$$

Note: Sync frames are transmitted in the static segment only. Thus $FID \leq 1023$.

51.6.15.2 Sync Frame Rejection Filtering

The synchronization frame rejection filter is a comparator. The compare value is defined by the [Sync Frame ID Rejection Filter Register \(FR_SFIDRFR\)](#). A received synchronization frame with the frame ID FID passes the sync frame rejection filter if [Equation 64](#) or [Equation 65](#) evaluates to true.

Equation 64 $FR_MCR[SFFE] = 0$

Equation 65 $FID \neq FR_SFIDRFR[SYNFRID]$

Note: Sync frames are transmitted in the static segment only. Thus $FID \leq 1023$.

51.6.16 Strobe signal support

The CC provides a number of strobe signals for observing internal protocol timing related signals in the protocol engine. The signals are listed and described in [Table 980](#).

51.6.16.1 Strobe signal assignment

Each of the strobe signals listed in [Table 980](#) can be assigned to one of the four strobe ports using the [Strobe Signal Control Register \(FR_STBSCR\)](#). To assign multiple strobe signals, the application must write multiple times to the [Strobe Signal Control Register \(FR_STBSCR\)](#) with appropriate settings.

To read out the current settings for a strobe signal with number n , the application must execute the following sequence.

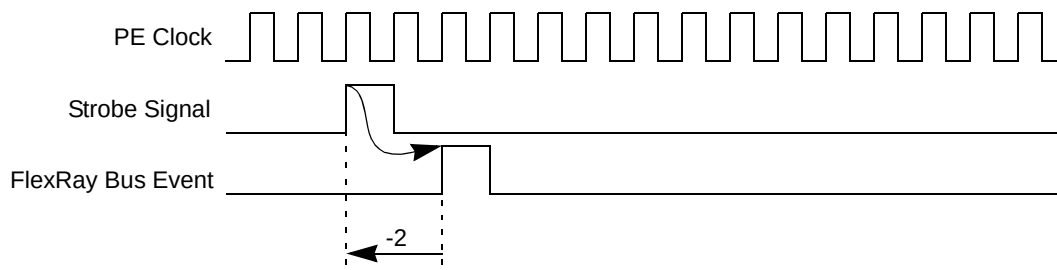
1. Write to FR_STBSCR with WMD = 1 and SEL = n . (Updates the SEL field only.)
2. Read STBCSR.
The SEL field provides n and the ENB and STBPSEL fields provides the settings for signal n .

51.6.16.2 Strobe signal timing

This section provides detailed timing information of the strobe signals with respect to the protocol engine clock.

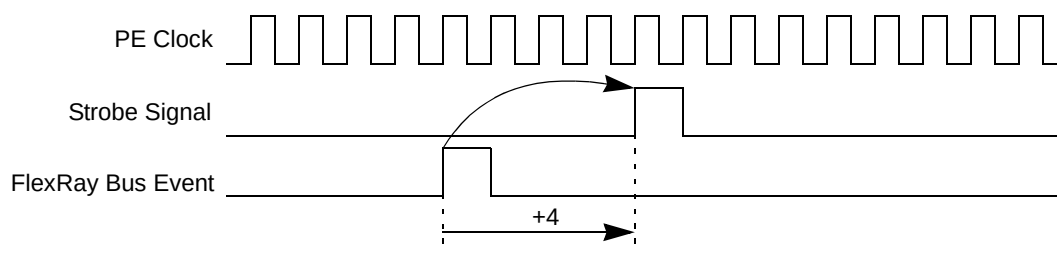
The strobe signals display internal PE signals. Due to the internal architecture of the PE, some signals are generated several PE clock cycles before the actual action is performed on the FlexRay bus. These signals are listed in [Table 980](#) with a negative clock offset. An example waveform is given in [Figure 1094](#).

Figure 1094. Strobe signal timing (type = pulse, clk_offset = -2)



Other signals refer to events that occurred on the FlexRay bus some cycles before the strobe signal is changed. These signals are listed in [Table 980](#) with a positive clock offset. An example waveform is given in [Figure 1095](#).

Figure 1095. Strobe signal timing (type = pulse, clk_offset = +4)



51.6.17 Timer support

The CC provides two timers that run on the FlexRay time base. Each timer generates a maskable interrupt when it reaches a configured point in time. Timer T1 is an absolute timer. Timer T2 can be configured to be an absolute or a relative timer. Both timers can be configured to be repetitive. In Non-repetitive mode, the timer stops if it expires. In Repetitive mode, the timer is restarted when it expires.

Both timers are active only when the protocol is in *POC:normal active* or *POC:normal passive* state. If the protocol is not in one of these modes, the timers are stopped. The application must restart the timers when the protocol has reached the *POC:normal active* or *POC:normal passive* state.

51.6.17.1 Absolute timer T1

Absolute timer T1 has the protocol cycle count and the macrotick count as the time base. The timer 1 interrupt flag TI1_IF in the [Protocol Interrupt Flag Register 0 \(FR_PIFR0\)](#) is set at the macrotick start event, if [Equation 66](#) and [Equation 67](#) are fulfilled.

Equation 66

$$\begin{aligned} & \text{CYCTR}[\text{CTCNT}] \& \text{FR_TI1CYSR}[\text{T1_CYC_MSK}] = \\ & = \text{FR_TI1CYSR}[\text{T1_CYC_VAL}] \& \text{FR_TI1CYSR}[\text{T1_CYC_MSK}] \end{aligned}$$

Equation 67
$$\text{FR_MTCTR}[\text{MTCT}] = \text{FR_TI1MTOR}[\text{T1_MTOFFSET}]$$

If the timer 1 interrupt enable bit `TI1_IE` in the [Protocol Interrupt Enable Register 0 \(FR_PIER0\)](#) is asserted, an interrupt request is generated.

The status bit `T1ST` is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the `T1ST` bit is not cleared and the timer is restarted immediately. The `T1ST` is cleared when the timer is stopped.

The corresponding interrupt condition (that is, when [Equation 66](#) and [Equation 67](#) are fulfilled) also leads to an event-out indication on an output port (`fr_evt_tim1`) based on the `[TIM1_EE]` bit of the [Protocol Event Output Enable and StopWatch Control Register \(FR_PEOER\)](#).

51.6.17.2 Absolute/relative timer T2

Timer T2 can be configured to be an absolute or relative timer by setting the `T2_CFG` control bit in the [Timer Configuration and Control Register \(FR_TICCR\)](#). The status bit `T2ST` is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the `T2ST` bit is not cleared and the timer is restarted immediately. The `T2ST` is cleared when the timer is stopped.

51.6.17.2.1 Absolute timer T2

If timer T2 is configured as an absolute timer, it has the same functionality as timer T1 but the configuration from [Timer 2 Configuration Register 0 \(FR_TI2CR0\)](#) and [Timer 2 Configuration Register 1 \(FR_TI2CR1\)](#) is used. When timer T2 expires, the interrupt flag `TI2_IF` in the [Protocol Interrupt Flag Register 0 \(FR_PIFR0\)](#) is set. If the timer 1 interrupt enable bit `TI1_IE` in the [Protocol Interrupt Enable Register 0 \(FR_PIER0\)](#) is asserted, an interrupt request is generated.

The corresponding interrupt condition for T2 also leads to an event out indication on an output port (`fr_evt_tim2`) based on the `[TIM2_EE]` bit of the [Protocol Event Output Enable and StopWatch Control Register \(FR_PEOER\)](#).

51.6.17.2.2 Relative Timer T2

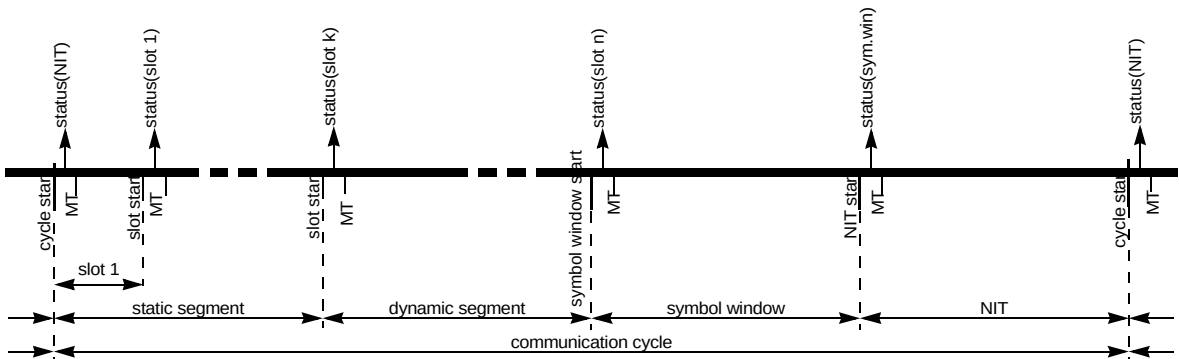
If timer T2 is configured as a relative timer, the interrupt flag `TI2_IF` in the [Protocol Interrupt Flag Register 0 \(FR_PIFR0\)](#) is set when the programmed amount of macroticks `MT[31:0]`—defined by [Timer 2 Configuration Register 0 \(FR_TI2CR0\)](#) and [Timer 2 Configuration Register 1 \(FR_TI2CR1\)](#)—has elapsed since the trigger or restart of timer T2. The relative timer is implemented as a down counter and expires when it has reached 0. At the macrotick start event, the value of `MT[31:0]` is checked and then decremented. Thus, if the timer is started with `MT[31:0] == 0`, it expires at the next macrotick start. The corresponding interrupt condition for T2 also leads to an event out indication on an output port (`fr_evt_tim2`) based on the `[TIM2_EE]` bit of the [Protocol Event Output Enable and StopWatch Control Register \(FR_PEOER\)](#).

51.6.18 Slot status monitoring

The CC provides several means for slot status monitoring. All slot status monitors use the same slot status vector provided by the PE. The PE provides a slot status vector for each static slot, for each dynamic slot, for the symbol window, and for the NIT, on a per-channel base. The content of the slot status vector is described in [Table 1094](#). The PE provides the

slot status vector within the first macrotick after the end of the related slot/window/NIT, as shown in [Figure 1096](#).

Figure 1096. Slot status vector update



Note: The slot status for the NIT of cycle n is provided after the start of cycle $n+1$.

Table 1094. Slot status content

	Status content
static / dynamic Slot	Slot-related status vSS!ValidFrame—valid frame received vSS!SyntaxError—syntax error occurred while receiving vSS!ContentError—content error occurred while receiving vSS!BViolation—boundary violation while receiving for slots in which the module transmits: vSS!TxConflict—reception ongoing while transmission starts for slots in which the module does not transmit: vSS!TxConflict—reception ongoing while transmission starts first valid - channel that has received the first valid frame Received-frame-related status extracted from a) header of valid frame, if vSS!ValidFrame = 1 b) last received header, if vSS!ValidFrame = 0 c) set to 0, if nothing was received vRF!Header!NFIndicator—null frame indicator (0 for null frame) vRF!Header!SuFIndicator—startup frame indicator vRF!Header!SyFIndicator—sync frame indicator

Table 1094. Slot status content (continued)

	Status content
Symbol window	Window-related status <i>vSS!ValidFrame</i> —always 0 <i>vSS!ContentError</i> —content error occurred while receiving <i>vSS!SyntaxError</i> —syntax error occurred while receiving <i>vSS!BViolation</i> —boundary violation while receiving <i>vSS!TxConflict</i> —reception ongoing while transmission starts Received-symbol-related status <i>vSS!ValidMTS</i> —valid Media Test Access Symbol received Received-frame-related status see static/dynamic slot
NIT	NIT-related status <i>vSS!ValidFrame</i> —always 0 <i>vSS!ContentError</i> —content error occurred while receiving <i>vSS!SyntaxError</i> —syntax error occurred while receiving <i>vSS!BViolation</i> —boundary violation while receiving <i>vSS!TxConflict</i> —always 0 Received-frame-related status see static/dynamic slot

51.6.18.1 Channel Status Error Counter Registers

The two channel status error counter registers, *Channel A Status Error Counter Register (FR_CASERCR)* and *Channel B Status Error Counter Register (FR_CBSECR)* are incremented by one if at least one of four slot status error bits, *vSS!SyntaxError*, *vSS!ContentError*, *vSS!BViolation*, or *vSS!TxConflict* is set to 1. The status vectors for all slots in the static and dynamic segment, in the symbol window, and in the NIT are taken into account. The counters wrap around after they have reached the maximum value.

51.6.18.2 Protocol Status Registers

The *Protocol Status Register 2 (FR_PSR2)* provides slot status information about the NIT and the symbol window. The *Protocol Status Register 3 (FR_PSR3)* provides aggregated slot status information.

51.6.18.3 Slot Status Registers

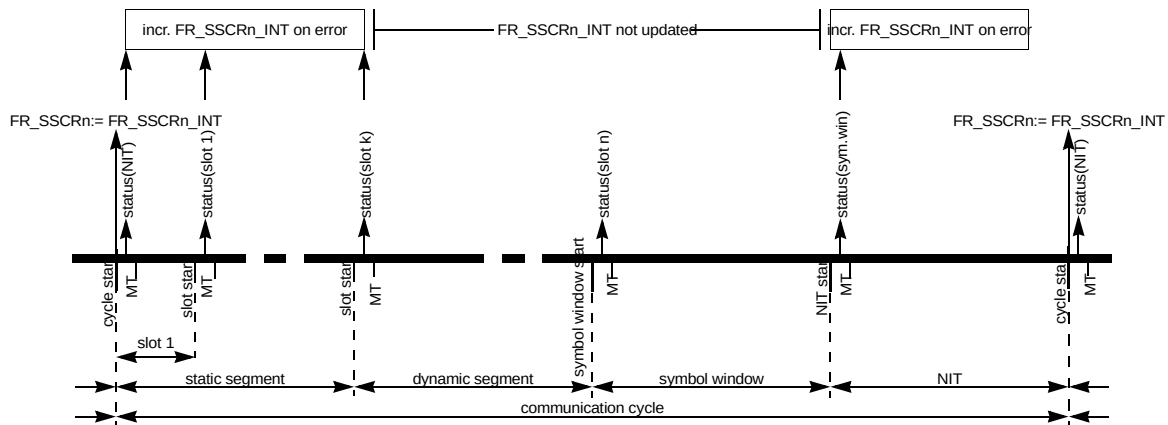
The eight slot status registers, *Slot Status Registers (FR_SSR0–FR_SSR7)*, can be used to observe the status of static slots, dynamic slots, the symbol window, or the NIT without individual message buffers. These registers provide all slot status-related and received frame/symbol-related status information, as given in *Table 1094*, except for the *first valid* indicator for non-transmission slots.

51.6.18.4 Slot Status Counter Registers

The CC provides four slot status error counter registers, *Slot Status Counter Registers (FR_SSCR0–FR_SSCR3)*. Each of these slot status counter registers is updated with the value of an internal slot status counter at the start of a communication cycle. The internal slot status counter is incremented if its increment condition, defined by the *Slot Status*

Counter Condition Register (FR_SSCCR), matches the status vector provided by the PE. All static slots, the symbol window, and the NIT status are taken into account. *Dynamic* slots are *excluded*. The internal slot status counting and update timing is shown in [Figure 1097](#).

Figure 1097. Slot status counting and FR_SSCCRn update



The PE provides the status of the NIT in the first slot of the next cycle. Due to these facts, the FR_SSCCRn register reflects, in cycle n , the status of the NIT of cycle $n - 2$, and the status of all static slots and the symbol window of cycle $n - 1$.

The increment condition for each slot status counter consists of two parts: the frame-related condition part and the slot-related condition part. The internal slot status counter FR_SSCCRn_INT is incremented if at least one of the following conditions is fulfilled:

1. Frame-related condition:

- $(\text{FR_SSCCRn}[\text{VFR}] \mid \text{FR_SSCCRn}[\text{SYF}] \mid \text{FR_SSCCRn}[\text{NUF}] \mid \text{FR_SSCCRn}[\text{SUF}]) \parallel$
count on frame condition
= 1;

and

- $(\sim \text{FR_SSCCRn}[\text{VFR}] \mid \text{vSS!ValidFrame}) \& \parallel$ valid frame restriction
 $(\sim \text{FR_SSCCRn}[\text{SYF}] \mid \text{vRF!Header!SyFIndicator}) \& \parallel$ sync frame indicator restriction
 $(\sim \text{FR_SSCCRn}[\text{NUF}] \mid \sim \text{vRF!Header!NFIndicator}) \& \parallel$ null frame indicator restriction
 $(\sim \text{FR_SSCCRn}[\text{SUF}] \mid \text{vRF!Header!SuFIndicator}) \parallel$ startup frame indicator restriction
= 1;

Note: The indicator bits SYF, NUF, and SUF are valid only when a valid frame was received. Thus, the VFR must always be set whenever the count-on-frame condition is used.

2. Slot-related condition:

- $(\text{FR_SSCCRn}[\text{STATUSMASK}[3]] \& \text{vSS!ContentError}) \parallel$ increment on content error
 $(\text{FR_SSCCRn}[\text{STATUSMASK}[2]] \& \text{vSS!SyntaxError}) \parallel$ increment on syntax error
 $(\text{FR_SSCCRn}[\text{STATUSMASK}[1]] \& \text{vSS!BViolation}) \parallel$ increment on boundary violation
 $(\text{FR_SSCCRn}[\text{STATUSMASK}[0]] \& \text{vSS!TxConflict}) \parallel$ increment on transmission conflict
= 1;

If the slot status counter is in Single-cycle mode (that is, FR_SSCCRn[MCY] = 0), the internal slot status counter FR_SSCCRn_INT is reset at each cycle start. If the slot status

counter is in Multicycle mode (that is, $FR_SSCCRn[MCY] = 1$), the counter is not reset and incremented until the maximum value is reached.

51.6.18.5 Message Buffer Slot Status field

Each individual message buffer and each FIFO message buffer provides a slot status field, which provides the information shown in [Table 1094](#) for the static/dynamic slot. The update conditions for the slot status field depend on the message buffer type. Refer to the Message Buffer Update sections in [Section 51.6.3.4.1, Individual message buffer configuration data](#).

51.6.19 System bus access

This section provides a description of the system bus access failures and the related CC behavior. System bus access failures may occur when the CC transfers data to or from the FlexRay memory area.

The system bus access failure types are described in [Section 51.6.19.1, "System bus access failure types"](#).

The behavior of the CC after the occurrence of a system bus access failure is described in [Section 51.6.19.2, "System bus access failure response"](#).

51.6.19.1 System bus access failure types

This section describes the two types of system bus access failures.

51.6.19.1.1 System bus illegal address access

A system bus illegal address access is detected when the CC has used an illegal or invalid address to access the FlexRay system memory area. There are three conditions that are treated as a system bus illegal address access:

- The system bus subsystem detects a CC access to an illegal system memory address.
- The CC detects the usage of a data field offset with the value of 0.
- The CC detects a memory error while reading a data field offset from the CHI LRAM memory (see [Section 51.6.24.3.1, "CHI LRAM error response after CC read"](#)).

If a system bus illegal address access is detected, the CC sets the ILSA_EF flag in the [CHI Error Flag Register \(FR_CHIERFR\)](#).

51.6.19.1.2 System bus access timeout

A system bus access timeout is detected if an access to the FlexRay memory area is not finished in time. The timeout value is derived from the SYMATOR[TIMEOUT] setting (see [Section 51.7.1.1, "Configure System Memory Access Timeout Register \(FR_SYMATOR\)"](#)).

If a system bus access timeout is detected, the CC sets the SBCF_EF flag in the [CHI Error Flag Register \(FR_CHIERFR\)](#).

51.6.19.2 System bus access failure response

This section describes the two types of behavior of the CC after the occurrence of a system bus access failure. The actual behavior is defined by the SBFF bit in the [Module Configuration Register \(FR_MCR\)](#).

51.6.19.2.1 Continue after system bus access failure

If FR_MCR[SBFF] is 0, the CC will continue its operation after the occurrence of the system bus access failure, but it will not generate any system bus accesses until the start of the next communication cycle. Since no data is read from or written to the FlexRay memory area, no messages are received or transmitted. Consequently, none of the individual message buffers or receive FIFOs will be updated until the next communication cycle starts.

If a frame is under transmission when the system bus failure occurs, a correct frame is generated in which the remaining header and frame data are replaced by all zeros. Depending on the point in time, this can affect the PPI bit, the header CRC, the payload length in the case of a dynamic slot, and the payload data. Starting from the next slot in the current cycle, no frames will be transmitted or received, except for the key slot; a sync or startup null-frame is transmitted if the key slot is assigned.

If a frame is received when the system bus failure occurs, the reception is aborted and the related receive message buffer is not updated.

Normal operation is resumed after the start of next communication cycle.

51.6.19.2.2 Freeze after system bus access failure

If FR_MCR[SBFF] is set to 1, the CC will go into Freeze mode immediately after the occurrence of one of the system bus access failures.

51.6.20 Interrupt support

The CC provides 172 individual interrupt sources and five combined interrupt sources.

51.6.20.1 Individual interrupt sources

51.6.20.1.1 Message buffer interrupts

The CC provides 128 message buffer interrupt sources.

Each individual message buffer provides an interrupt flag FR_MBCCSRn[MBIF] and an interrupt enable bit FR_MBCCSRn[MBIE]. The CC sets the interrupt flag when the slot status of the message buffer was updated. If the interrupt enable bit is asserted, an interrupt request is generated.

51.6.20.1.2 FIFO interrupts

The CC provides two FIFO interrupt sources.

Each of the two FIFOs provides a Receive FIFO Almost Full Interrupt Flag. The CC sets the Receive FIFO Almost Full Interrupt Flags (FR_GIFER[FAFBIF], FR_GIFER[FAFAIF]) in the [Global Interrupt Flag and Enable Register \(FR_GIFER\)](#) if the corresponding receive FIFO fill level exceeds the defined watermark.

51.6.20.1.3 Wakeup interrupt

The CC provides one interrupt source related to the wakeup.

The CC sets the Wakeup Interrupt Flag FR_GIFER[WUPIF] when it has received a wakeup symbol on the FlexRay bus. The CC generates an interrupt request if the interrupt enable bit FR_GIFER[WUPIE] is asserted.

51.6.20.1.4 Protocol interrupts

The CC provides 25 interrupt sources for protocol-related events. For details, see the [Protocol Interrupt Flag Register 0 \(FR_PIFR0\)](#) and [Protocol Interrupt Flag Register 1 \(FR_PIFR1\)](#). Each interrupt source has its own interrupt enable bit.

51.6.20.1.5 CHI interrupts

The CC provides 16 interrupt sources for CHI-related error events. For details, see the [CHI Error Flag Register \(FR_CHIERFR\)](#). There is one common interrupt enable bit FR_GIFER[CHIE] for all CHI error interrupt sources.

51.6.20.2 Combined interrupt sources

Each combined interrupt source generates an interrupt request only when at least one of the interrupt sources that is combined generates an interrupt request.

51.6.20.2.1 Receive message buffer interrupt

The receive message buffer interrupt request is generated when at least one of the individual receive message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[RBIE] is set.

51.6.20.2.2 Transmit message buffer interrupt

The transmit message buffer interrupt request is generated when at least one of the individual transmit message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[TBIE] is asserted.

51.6.20.2.3 Protocol interrupt

The protocol interrupt request is generated when at least one of the individual protocol interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[PRIE] is set..

51.6.20.2.4 CHI interrupt

The CHI interrupt request is generated when at least one of the individual chi error interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[CHIE] is set.

51.6.20.2.5 Module interrupt

The module interrupt request is generated if at least one of the combined interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[MIE] is set.

Figure 1098. .Scheme of FR_GIFER interrupt signal generation

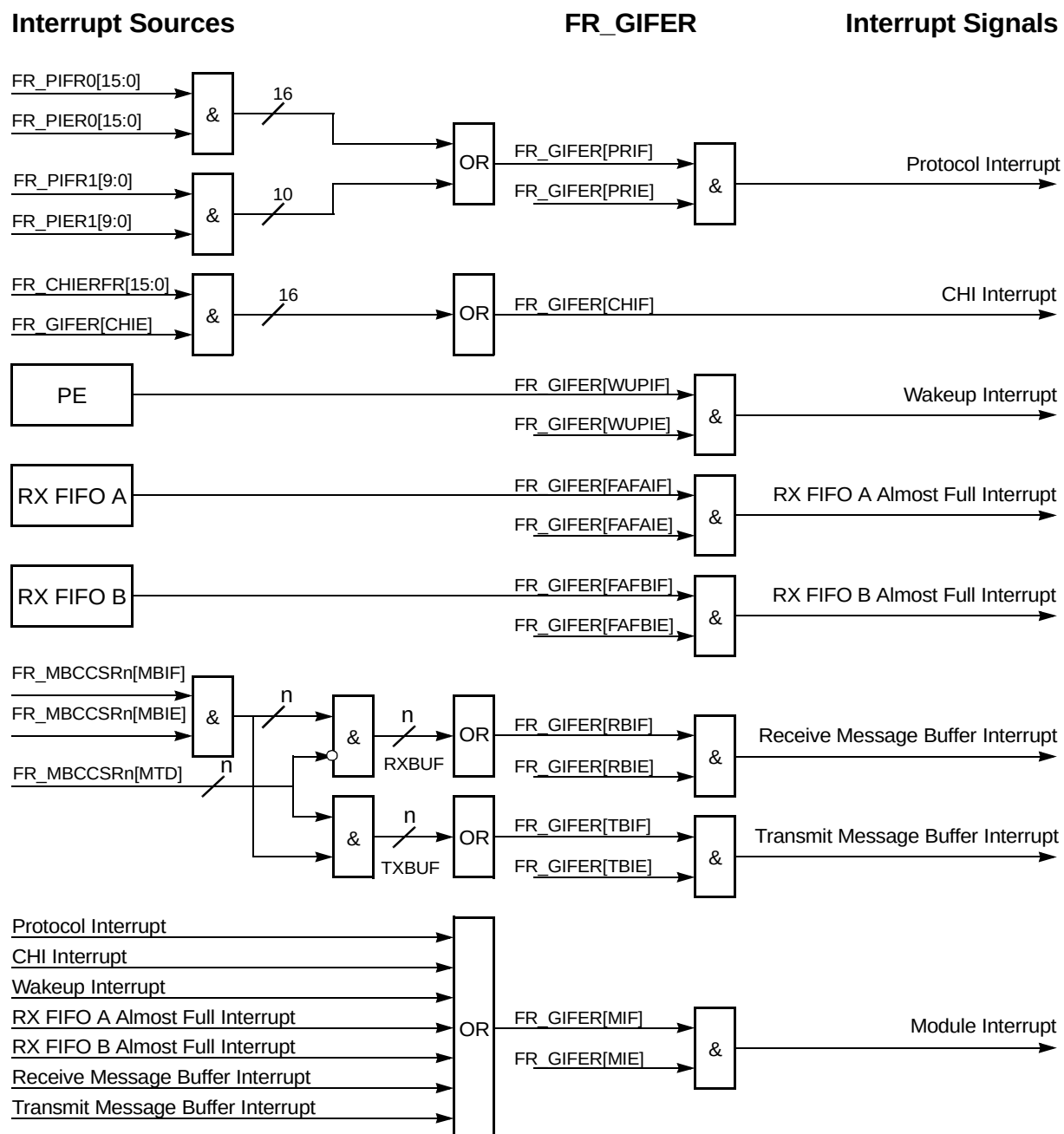


Figure 1099. FR_EEIFER interrupt signal generation scheme

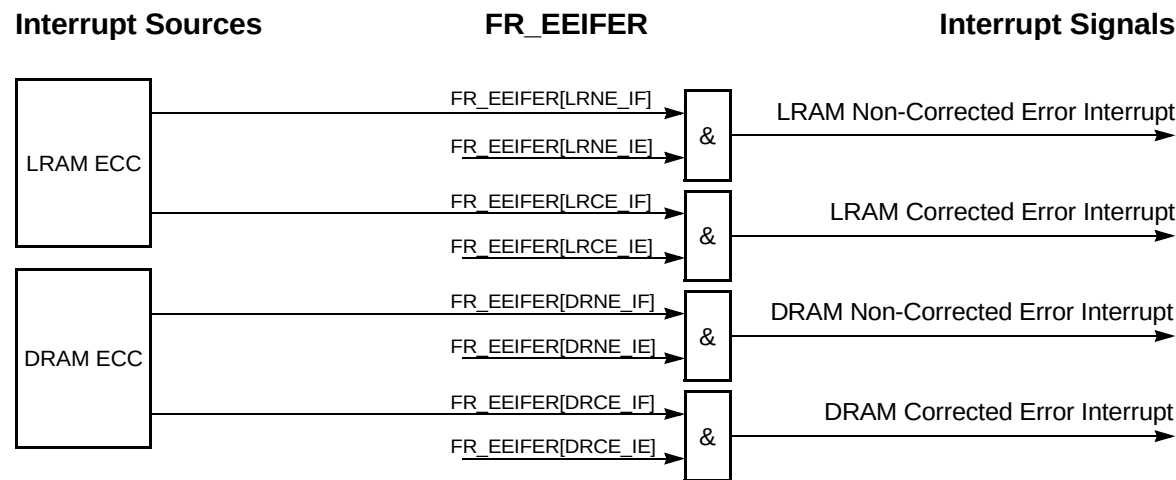
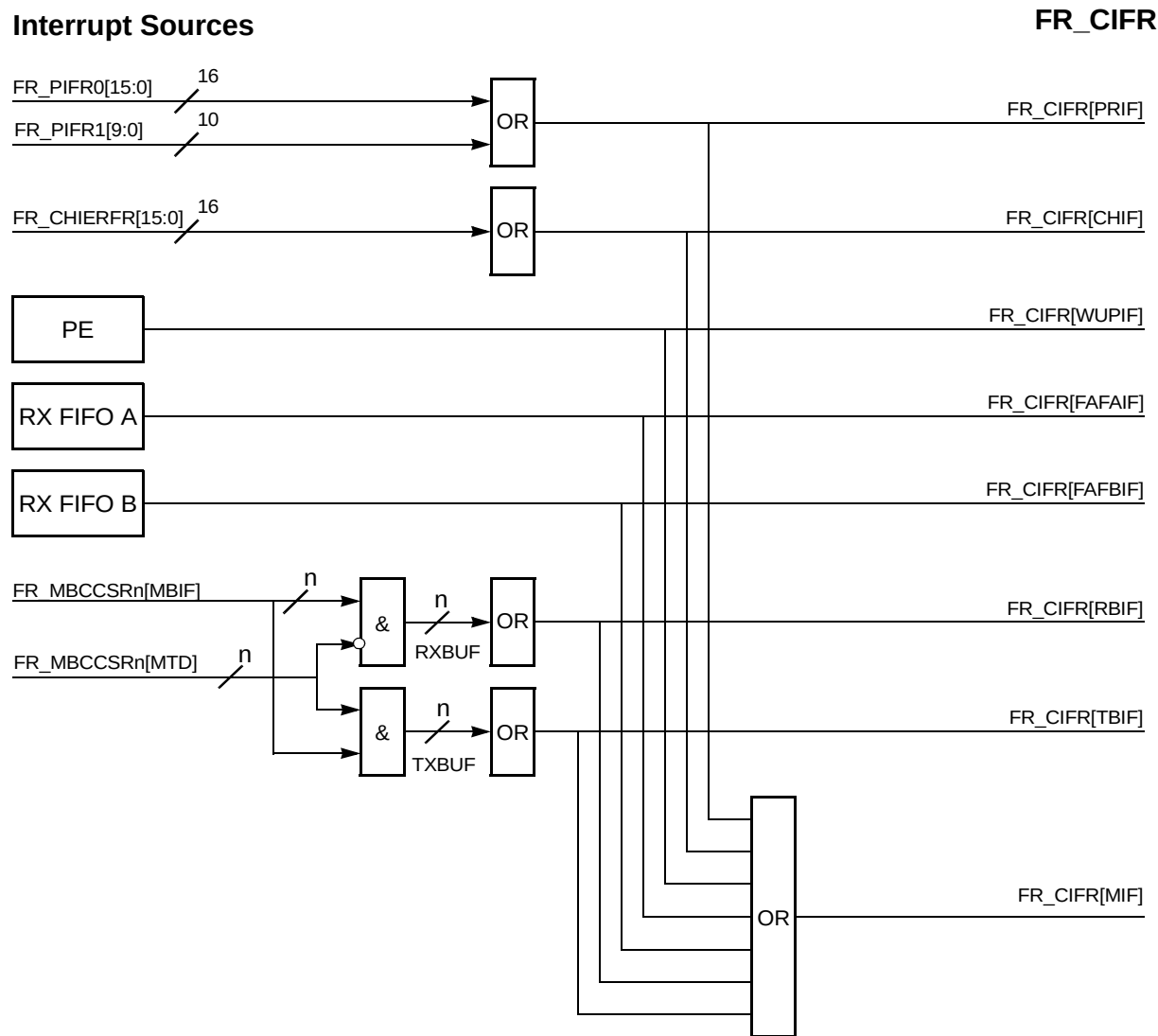


Figure 1100. FR_CIFR flag generation scheme



51.6.21 Lower bit rate support

The CC supports a number of lower bit rates on the FlexRay bus channels. The lower bit rates are implemented by modifying the duration of the microtick *pdMicrotick*, the number of samples per microtick *pSamplesPerMicrotick*, the number of samples per bit *cSamplesPerBit*, and the strobe offset *cStrobeOffset*. The application configures the FlexRay channel bit rate by setting the BITRATE field in the *Module Configuration Register (FR_MCR)*. The protocol values are set internally. The available bit rates, the related BITRATE field configuration settings, and the related protocol parameter values are shown in *Table 1095*.

Table 1095. FlexRay channel bit rate control

FlexRay channel bit rate [Mbit/s]	FR_MCR[BITRATE]	<i>pdMicrotick</i> [ns]	<i>gdSampleClockPeriod</i> [ns]	pSamplesPerMicrotick	cSamplesPerBit	cStrobeOffset
10.0	000	25.0	12.5	2	8	5
8.0	011	25.0	12.5	2	10	6
5.0	001	25.0	25.0	1	8	5
2.5	010	50.0	50.0	1	8	5

Note: The bit rate of 8 Mbit/s is not defined by the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

51.6.22 PE data memory (PE DRAM)

The PE data memory (PE DRAM) is 128 word, 16-bit wide memory with byte access that contains the program data of the PE internal CPU. The PE DRAM is divided into two banks of 8 bits each. The memory data [7:0] is assigned to BANK0 and the memory data [15:8] is assigned to BANK1.

Table 1096. PE DRAM layout

ADDR	BANK1	BANK0
0x00	byte1	byte0
0x01	byte3	byte2
...		
0x7F	byte255	byte254

The FlexRay module provides means to access the PE DRAM from the application. The PE DRAM application access is initiated and controlled via [PE DRAM Access Register \(FR_PEDRAR\)](#) and [PE DRAM Access Register \(FR_PEDRAR\)](#). This functionality is used to check the memory error detection.

51.6.22.1 PE DRAM read access

A read access from the PE DRAM can be initiated in any protocol state. The following sequence describes a read access from the PE DRAM address 0x70.

1. FR_PEDRAR:= 0x50E0;
// INST=0x5; ADDR=070
2. Wait until FR_PEDRAR[DAD] == 1;
// wait for end of PE DRAM access
3. val = FR_PEDRDR[DATA];
// read PE DRAM data

The read access is handled by the PE internal CPU with the lowest execution priority. This may cause a response delay with a maximum of 1000 PE clock cycle (25 µs).

51.6.22.2 PE DRAM write access

A write access into the PE DRAM can be initiated in any protocol state. The following sequence describes a write access to the PE DRAM address 0x70.

1. FR_PEDRDR:= DATA;
// write value to be written into data register
2. FR_PEDRAR:= 0x30E0;
// INST=0x3; ADDR=0x70
3. wait until FR_PEDRAR[DAD] == 1;
// wait for end of PE DRAM access
4. val = FR_PEDRDR[DATA];
// read back PE DRAM data

The write access is handled by the PE internal CPU with the lowest execution priority. This may cause an response delay with a maximum of 1000 PE clock cycle (25 µs).

If the conditions given in [Section 51.6.22.3, “PE DRAM write access limitations](#) are fulfilled, the data provided in [PE DRAM Access Register \(FR_PEDRAR\)](#) is written into the PE DRAM, read back in the next clock cycle, and stored into the [PE DRAM Access Register \(FR_PEDRAR\)](#). Otherwise, data is not written into the PE DRAM and 0x0000 is stored into the [PE DRAM Access Register \(FR_PEDRAR\)](#).

51.6.22.3 PE DRAM write access limitations

The PE DRAM is used by the protocol engine if the module is not in *POC:default config* state. The only address not used by the protocol engine is 0x70. To prevent the corruption of protocol engine data the following PE DRAM write access limitations apply for application writes.

1. When the module is in *POC:default config* state, all PE DRAM addresses are writable.
2. When the module is not in *POC:default config* state, only PE DRAM address 0x70 is writable.

51.6.23 CHI lookup-table memory (CHI LRAM)

The CHI lookup-table memory (CHI LRAM) is an CHI-internal memory that contains the message buffer configuration data and the data field offsets for the physical message buffers. The configuration data for two message buffers or six data field offsets are contained in one memory row. The CHI LRAM is divided into six memory banks.

Table 1097. CHI LRAM layout

ADR	BANK5	BANK4	BANK3	BANK2	BANK1	BANK0
0x00	FR_MBIDXR1	FR_MBFIDR1	FR_MBCCFR1	FR_MBIDXR0	FR_MBFIDR0	FR_MBCCFR0
0x01	FR_MBIDXR3	FR_MBFIDR3	FR_MBCCFR3	FR_MBIDXR2	FR_MBFIDR2	FR_MBCCFR2
...						
0x3F	FR_MBIDXR127	FR_MBFIDR127	FR_MBCCFR127	FR_MBIDXR126	FR_MBFIDR126	FR_MBCCFR126
0x40	FR_MBDOR5	FR_MBDOR4	FR_MBDOR3	FR_MBDOR2	FR_MBDOR1	FR_MBDOR0
...						
0x55	FR_MBDOR131	FR_MBDOR130	FR_MBDOR129	FR_MBDOR128	FR_MBDOR127	FR_MBDOR126
0x56	FR_LEETR5	FR_LEETR4	FR_LEETR3	FR_LEETR2	FR_LEETR1	FR_LEETR0

51.6.23.1 CHI LRAM read and write access

The CHI LRAM is accessed by the application via regular register read and write accesses.

51.6.24 Memory content error detection

The FlexRay module provides integrated memory content error detection for both the CHI LRAM and PE DRAM, and memory content error correction for the PE DRAM. The memory error detection for the CHI LRAM uses a standard Hamming code with a Hamming distance of 3 and detects all single-bit and double-bit errors (SEDED). The memory error detection and correction for the PE DRAM uses an enhanced Hamming code with a Hamming distance of 4 and detects and corrects all single-bit errors and detects all double-bit errors (SEDED).

This section describes the reporting of the occurrence of memory content errors, the reaction of the module on the occurrence, and how the application can inject memory errors in order to trigger the report and response behavior.

51.6.24.1 Memory error types

A memory error is the distortion of one or more bits read out of the memory. The reading of the values of all zeros and all ones is considered as a special case. The FlexRay module detects and indicates the memory errors as shown in [Table 1098](#). The entries on the top have higher priority.

Each memory read access reads out *all* banks of the addressed row and runs error detection on *all* banks, even in the case that the application has triggered a read from only one bank. This may lead to the reporting of an memory error if at least one bank contains a memory error, even if an error-free bank has been read.

Table 1098. Detected memory error types

Memory	Priority	Memory data	Indication
CHI LRAM	0 (highest)	All zeros	Non-corrected error
PE DRAM			Non-corrected error
CHI LRAM		All ones	Non-corrected error
PE DRAM			Non-corrected error
CHI LRAM	1 (lowest)	One bit flipped	Non-corrected error
PE DRAM			Corrected error
CHI LRAM		Two bits flipped	Non-corrected error
PE DRAM			Non-corrected error
CHI LRAM		Three or more bits flipped	One out of {no error, non-corrected error}, defined by coding given in Section 51.6.24.2.3: CHI LRAM checkbits and Section 51.6.24.2.4: CHI LRAM syndrome
PE DRAM			One out of {no error, corrected error, non-corrected error}, defined by coding given in Section 51.6.24.2.1: PE DRAM checkbits and Section 51.6.24.2.2: PE DRAM syndrome

51.6.24.2 Memory error reporting

The memory error reporting is enabled only if the ECC functionality enable bit ECCE in the [Module Configuration Register \(FR_MCR\)](#) is set.

For each of the two memories, there are two sets of internal registers to store the detection of one corrected and one non-corrected memory error.

If a memory error is detected, the module checks whether the related error interrupt flag in the [ECC Error Interrupt Flag and Enable Register \(FR_EEIFER\)](#) is set.

- If the error interrupt flag is set, the related internal error reporting register is not updated and the related error overflow flag is set to 1 to indicate a loss of error condition.
- If the error interrupt flag is not set, the internal reporting register is updated and the error interrupt flag is set to 1. If two or more memory errors of the same type are detected, the error for the bank with the lower bank number will be reported, and the error overflow flag will be set to 1.

If a memory error is detected for at least two banks of one memory, the related error overflow flag is set to 1 to indicate a loss of error condition.

In addition to the above, the error indications (corrected/non-corrected) along with the failing address are sent out from the module for external error logging and corrective actions by the software.

51.6.24.2.1 PE DRAM checkbits

The coding of the checkbits reported in [ECC Error Report Code Register \(FR_EECCR\)](#) for PE DRAM memory errors is shown in [Table 1098](#). This table shows the implemented enhanced Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 1099. PE DRAM checkbits coding

CODE	CODE				DATA							
	3	2	1	0	7	6	5	4	3	2	1	0
4 ⁽¹⁾	X	X	X	X	X	X	X	X	X	X	X	X
3 ⁽²⁾	—	—	—	—	X	X	X	X	—	—	—	—
2	—	—	—	—	X	—	—	—	X	X	X	—
1	—	—	—	—	—	X	X	—	X	X	—	X
0	—	—	—	—	—	X	—	X	X	—	X	X

1. The checkbit CODE[4] is set to 1 if and only if there is an even number of 1's in columns with X.

2. The checkbits CODE[3]... CODE[0] are set to 1 if and only if there is an odd number of 1's in all columns with X.

This coding of the checkbit ensures that neither 0x000 nor 0xFF are valid code words written into the memory.

51.6.24.2.2 PE DRAM syndrome

The coding of the syndrome reported in the [ECC Error Report Code Register \(FR_EERCRCR\)](#) for PE DRAM memory errors is shown in [Table 1100](#).

Table 1100. FR_EERCRCR[CODE] PE DRAM syndrome coding

FR_EERCRCR[CODE]		Description
[4]	[3:0]	
0x1	0x0	No Error (never appears in error report registers)
0x0	0x0	If data == 0: Non Corrected Error (Dedicated Handling of All Zero Code Word) If data != 0: Corrected Error (Parity Bit 4)
0x0	0x1	Corrected error (parity bit 0)
0x0	0x2	Corrected error (parity bit 1)
0x0	0x3	Corrected error (data bit 0)
0x0	0x4	Corrected error (parity bit 2)
0x0	0x5	Corrected error (data bit 1)
0x0	0x6	Corrected error (data bit 2)
0x0	0x7	Corrected error (data bit 3)
0x0	0x8	Corrected error (parity bit 3)
0x0	0x9	Corrected error (data bit 4)
0x0	0xA	Corrected error (data bit 5)
0x0	0xB	Corrected error (data bit 6)
0x0	0xC	Corrected error (data bit 7)
0x0	0xD–0xF	Non-corrected error
0x1	0x1–0xF	Non-corrected error

51.6.24.2.3 CHI LRAM checkbits

The coding of the checkbits reported in [ECC Error Report Code Register \(FR_EECCR\)](#) for CHI LRAM memory errors is shown in [Table 1101](#). This table shows the implemented Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 1101. CHI LRAM checkbits coding

CODE ⁽¹⁾	DATA															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	X	X	X	X	X	—	—	—	—	—	—	—	—	—	—	—
3	—	—	—	—	—	X	X	X	X	X	X	X	—	—	—	—
2	X	X	—	—	—	X	X	X	X	—	—	—	X	X	X	—
1	—	—	X	X	—	X	X	—	—	X	X	—	X	X	—	X
0	X	—	X	—	X	X	—	X	—	X	—	X	X	—	X	X

1. The checkbit CODE[n] is set to 1 if and only if there is an odd number of 1's in all columns with X.

51.6.24.2.4 CHI LRAM syndrome

The coding of the syndrome reported in [ECC Error Report Code Register \(FR_EECCR\)](#) for CHI LRAM memory errors is shown in [Table 1102](#).

Table 1102. FR_EECCR[CODE] CHI LRAM syndrome coding

FR_EECCR[CODE]	Description
0x00	No Error (never appears in error report registers)
0x01–0x1F	Non-corrected error

51.6.24.3 Memory error response

The memory error response is enabled only when the ECC functionality enable bit ECCE in the [Module Configuration Register \(FR_MCR\)](#) is set.

In case a *corrected* memory error is detected, the FlexRay module continues its normal operation using the corrected data word. This section describes the behavior of the FlexRay module after the detection of a *non-corrected* memory error.

51.6.24.3.1 CHI LRAM error response after CC read

When the CC is out of the [POC:default config](#) state, it reads the configuration data and the data field offsets of all utilized message buffers in every slot and in the NIT. If a non-corrected memory error is detected during this module read access, the error response of the module depends from LRAM location where the error occurred.

- If the LRAM address belongs to physical message buffer configuration data the FlexRay module will consider the affected message buffer as disabled for the current

search and will exclude this buffer from the search. The configuration of the affected message buffer is not changed.

If the affected message buffer is a Tx message buffer, no frame will be transmitted from this message buffer in the next slot. If the affected message buffer is an Rx message buffer, no frame will be received to this message buffer in the next slot.

- If the LRAM address belongs to the data field offset area and the related physical message buffer is used for Rx or Tx, the first access to the system memory caused by payload read or write yields to the assertion of the FR_CHIERFR[ILSA_EF]. No memory access occurs with respect to payload access for the complete frame.

51.6.24.3.2 CHI LRAM error response after application read

The application can read the content of the CHI LRAM by reading the FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn, FR_MBDORn, and FR_LEETRn registers. If a non-corrected memory error is detected during this kind of read access, the module indicates the detected memory error, delivers the non-corrected data read, and continues its normal operation.

51.6.24.3.3 PE DRAM error response after CC read

If the CC detects a non-corrected memory error during an internal read of program data contained in PE DRAM, this is considered a fatal protocol error and the module enters the protocol freeze state immediately.

51.6.24.3.4 PE DRAM error response after application read in *POC:default config* state

If the CC detects a non-corrected memory error during an application-triggered read from any PE DRAM address and the protocol is in the *POC:default config* state, this is considered a fatal protocol error and the module enters the protocol freeze state. This behavior allows for checking the freeze functionality in case of the detection of non-corrected errors.

51.6.24.3.5 PE DRAM error response after application read out of *POC:default config*

If the CC detects a non-corrected memory error during an application-triggered read from any PE DRAM address and the protocol is not in the *POC:default config* state, this error is not considered a fatal error and the protocol state is not changed. This prevents any interference of the running protocol by PE DRAM error injection reads.

51.6.25 Memory error injection

The error injection functionality is used by the application to inject data errors into the memories to trigger and check the memory error detection functionality.

The error injection is enabled only if the ECC functionality enable bit ECCE in the *Module Configuration Register (FR_MCR)* and the error injection enable control bit EIE in the *ECC Error Report and Injection Control Register (FR_EERICR)* are set.

Error Injection mode is configured by the EIM configuration bit in the *ECC Error Report and Injection Control Register (FR_EERICR)*. When error injection is enabled, each write access to the configured memory location will be distorted.

The injector has the same behavior for FlexRay module memory writes and application memory writes.

51.6.25.1 CHI LRAM error injection

The following sequence describes an memory error injection sequence for the CHI LRAM memory. This sequence consists of the error injector setup followed by an application triggered write access to provoke an distortion of the memory content. The content of the CHI LRAM is described in [Table 1097](#).

When the CC is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see [Section 51.7.3, "Memory error injection out of POC:default config"](#).

Injector setup:

1. FR_MCR[ECCE]:= 1;
// enable ECC functionality
2. FR_EERICE[EIE]:=I_MODE;
// configure error injection mode
3. FR_EEIAR[MID]:= 1;
// select CHI LRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;
// select bank for error injection; I_BANK = {0,1,2,3,4,5}
5. FR_EEIAR[ADDR]:= I_ADDR;
// select address for error injection; I_ADDR ≤ 0x56
6. FR_EEIDR[DATA]:= D_DIST;
// define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;
// define checkbit distortion pattern
8. FR_EERICE[EIE]:=1;
// enable error injection

Application Write Access:

```
If (I_BANK==0) -> FR_MBCCFR(2n) / FR_MBDOR(6k) / FR_LEETR0 := DATA;
If (I_BANK==1) -> FR_MBFIDR(2n) / FR_MBDOR(6k+1) / FR_LEETR1 := DATA;
If (I_BANK==2) -> FR_MBIDXR(2n) / FR_MBDOR(6k+2) / FR_LEETR2 := DATA;
If (I_BANK==3) -> FR_MBCCFR(2n+1) / FR_MBDOR(6k+3) / FR_LEETR3 := DATA;
If (I_BANK==4) -> FR_MBFIDR(2n+1) / FR_MBDOR(6k+4) / FR_LEETR4 := DATA;
If (I_BANK==5) -> FR_MBIDXR(2n+1) / FR_MBDOR(6k+5) / FR_LEETR5 := DATA;
// write DATA to the defined injection bank and injection address (see Table 1097).
```

51.6.25.2 PE DRAM error injection

The following sequence describes a memory error injection sequence for the PE DRAM memory. This sequence consists of the error injector setup followed by an application triggered write access to provoke an distortion of the memory content.

When the FlexRay module is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see [Section 51.7.3.2, "PE DRAM error injection out of POC:default config"](#).

Injector setup:

1. FR_MCR[ECCE]:= 1;
// enable ECC functionality
2. FR_EERICE[EIE]:=I_MODE;
// configure error injection mode
3. FR_EEIAR[MID]:= 0;
// select PE DRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;
// define bank for error injection; I_BANK = {0,1}
5. FR_EEIAR[ADDR]:= I_ADDR;
// define address for error injection; I_ADDR ≤ 0x7F
6. FR_EEIDR[DATA]:= D_DIST;
// define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;
// define checkbit distortion pattern
8. FR_EERICE[EIE]:=1;
// enable error injection

Application write access (for example, I_ADDR=0x70):

1. FR_PEDRAR:= 0x30E0;
// INST=0x3; ADDR=0x70
2. wait until FR_PEDRAR[DAD] == 1;
// wait for end of PE DRAM access
3. val = FR_PEDRDR[DATA]; |
// get read back PE DRAM data

Note: The write access to the PE DRAM triggers an subsequent read access from PE DRAM in the next cycle, which triggers the detection of the distorted data.

51.6.26 StopWatch function

FlexRay supports a StopWatch function to facilitate OS synchronization to the FlexRay network. This provides a capability to capture the time difference in system clocks between the operating system timer tick and the FlexRay time base. This allows the software synchronization handler to adjust the OS clock to sync with the FR macrotick.

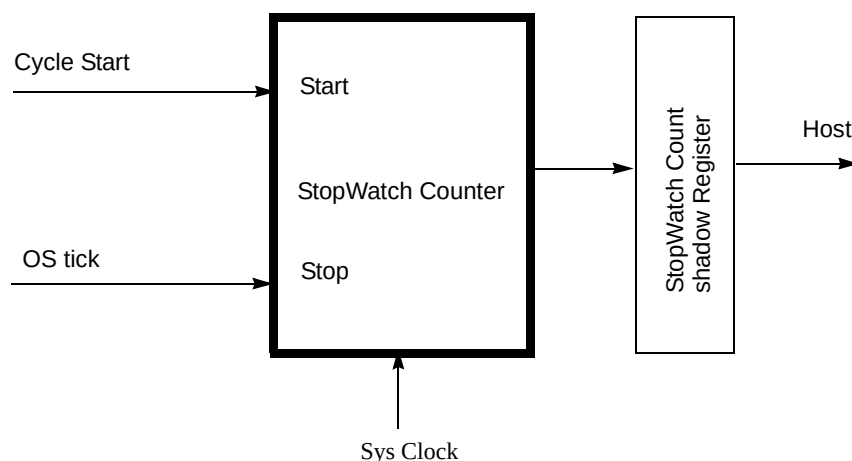


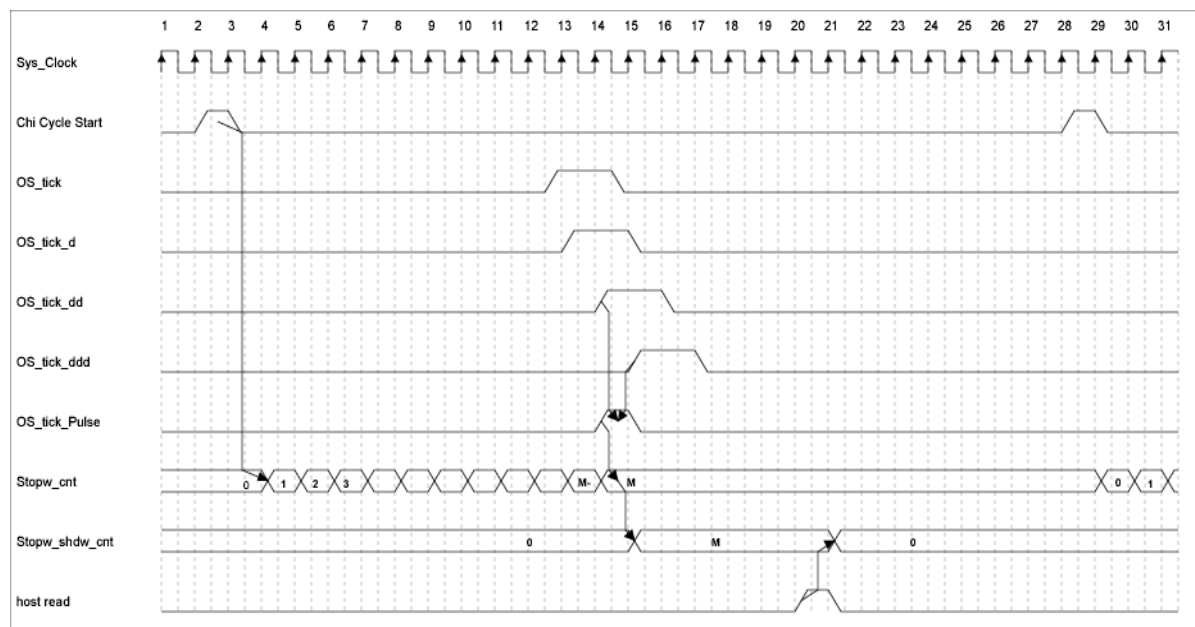
Figure 1101. StopWatch counter logic

As shown here, there is a 32 bit stopwatch counter that resets and starts with each cycle start. This counter runs on the system clock and increments every cycle until an OS tick event is received, after which the counter halts and the count value is transferred to a shadow register. The shadow register (*Stop Watch Count Register (FR_STPWR)*) holds the counter value till the host reads all the 32 bits, after which it automatically gets cleared.

The StopWatch logic is controlled through a control bit STPW_EN in the FR_PEOER (*Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)*). The counter runs only when this bit is set.

Figure 1102 shows the actual timing of the stopwatch counter logic operation.

Figure 1102. StopWatch function timing diagram



Note: 'Host read' implies that all 32 bits have been read out completely. Any partial read (i.e for 8/16 bits access) will not clear the StopWatch count register.

Note: "If cycle start and OS tick occurs simultaneously, timer will remain at 0. But OS_tick has a 2 stage synchronizer and cycle_start has a single stage synchronizer into chi clock domain. So simultaneously here means OS_tick_pulse and chi_cycle_start occurring at the same cycle."

51.7 Application information

51.7.1 Module configuration

This section describes essential parts of the module configuration.

51.7.1.1 Configure System Memory Access Timeout Register (FR_SYMATOR)

To ensure reliable operation of the CC, the application must ensure that the TIMEOUT value in the [System Memory Access Time-Out Register \(FR_SYMATOR\)](#) and the CHI clock frequency f_{CHI} in MHz fulfill [Equation 68](#)^(s).

Equation 68 $0 \leq \text{SYMATOR}[\text{TIMEOUT}] \leq \lfloor 0.45 \cdot f_{\text{CHI}} - 8 \rfloor$

For a given SYMATOR[TIMEOUT] value, f_{CHI} can be increased without causing unreliable operation of the CC. The same holds for reducing the SYMATOR[TIMEOUT] value for a given f_{CHI} .

Some examples for maximum values of the SYMATOR[TIMEOUT] for a minimum CHI frequency are given in [Table 1103](#).

Table 1103. Maximum SYMATOR[TIMEOUT] examples

f_{CHI}	SYMATOR[TIMEOUT]	f_{CHI}	SYMATOR[TIMEOUT]
$\geq 18 \text{ MHz}$	0	$\geq 100 \text{ MHz}$	≤ 37
$\geq 23 \text{ MHz}$	≤ 2	$\geq 120 \text{ MHz}$	≤ 46
$\geq 27 \text{ MHz}$	≤ 4	$\geq 140 \text{ MHz}$	≤ 55
$\geq 32 \text{ MHz}$	≤ 6	$\geq 160 \text{ MHz}$	≤ 64
$\geq 60 \text{ MHz}$	≤ 19	$\geq 180 \text{ MHz}$	≤ 73
$\geq 80 \text{ MHz}$	≤ 28	$\geq 200 \text{ MHz}$	≤ 82

51.7.1.1.1 System bus wait state constraints

The SYMATOR[TIMEOUT] value corresponds directly to a certain acceptable number of wait states on the system bus.

For single-channel configurations and situations in which the sync frame table generation functionality is *not* used ($\text{FR_SFTCCSR}[\text{SDVEN}, \text{SIDEN}] = 0$), no timeout will be detected if fewer than $2 \times \text{SYMATOR}[\text{TIMEOUT}] + 1$ wait states will be seen on the system bus for each system bus access.

For dual-channel configurations, or situations in which the sync frame table generation functionality is used, no timeout will be detected if fewer than $\text{SYMATOR}[\text{TIMEOUT}] - 1$ wait states will be seen on the system bus for each system bus access.

s. See [Section 51.3, "Controller Host Interface clocking"](#) for all constraints of minimum CHI clock frequency.

51.7.1.2 Configure data field offsets

The data field offsets are located in the [Message Buffer Data Field Offset Registers \(FR_MBDORn\)](#) and [Receive FIFO Start Data Offset Register \(FR_RFSDOR\)](#). The application has to configure the data field offset values for all message buffers that are used.

When the module is enabled, the initialization value of the FR_MBDORn[MBDO] and FR_RFSDOR[SDO] is 0. This value is considered to be illegal (see [Section 51.6.19.1.1, "System bus illegal address access"](#)).

51.7.2 Initialization sequence

This section describes the steps required to initialize the CC. The first subsection describes the steps required after a system reset; the second section describes the steps required after preceding shutdown of the CC.

51.7.2.1 Module initialization

This section describes the module-related initialization steps after a system reset.

1. Configure the CC.
 - a) Configure the control bits in the [Module Configuration Register \(FR_MCR\)](#).
 - b) Configure the system memory base address in the [System Memory Base Address Register \(FR_SYMBADR\)](#).
2. Enable the CC.
 - a) Write 1 to the module enable bit MEN in the [Module Configuration Register \(FR_MCR\)](#).

The CC now enters Normal mode. The application can commence with the protocol initialization described in [Section 51.7.2.2, "Protocol initialization"](#).

51.7.2.2 Protocol initialization

This section describes the protocol-related initialization steps.

1. Configure the Protocol Engine.
 - a) Issue CONFIG command via the [Protocol Operation Control Register \(FR_POCR\)](#).
 - b) Wait for *POC:config* in the [Protocol Status Register 0 \(FR_PSR0\)](#).
 - c) Configure the FR_PCR0,..., FR_PCR30 registers to set all protocol parameters.
2. Configure the message buffers and FIFOs.
 - a) Set the number of message buffers used and the message buffer segmentation in the [Message Buffer Segment Size and Utilization Register \(FR_MBSSUTR\)](#).
 - b) Define the message buffer data size in the [Message Buffer Data Size Register \(FR_MBDSR\)](#).
 - c) Configure each message buffer by setting the configuration values in the [Message Buffer Configuration, Control, Status Registers \(FR_MBCCSRn\)](#), [Message Buffer Cycle Counter Filter Registers \(FR_MBCCFRn\)](#), [Message Buffer Frame ID Registers \(FR_MBFIDRn\)](#), and [Message Buffer Index Registers \(FR_MBIDXRn\)](#).
 - d) Configure the FIFOs.
 - e) Issue CONFIG_COMPLETE command via the [Protocol Operation Control Register \(FR_POCR\)](#).
 - f) Wait for *POC:ready* in the [Protocol Status Register 0 \(FR_PSR0\)](#).

After this sequence, the CC is configured as a FlexRay node and is ready to integrate into the FlexRay cluster.

51.7.2.3 CHI LRAM initialization

The CHI LRAM initialization is performed by the CC when it leaves Disabled mode. The initialization runs for 87 CHI clock cycles. All fields in the FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBDORn, and LEETRn registers are initialized to 0. All application read or write accesses to these registers are delayed until the initialization is finished.

51.7.2.4 PE DRAM initialization

The PE DRAM initialization is performed by the CC in the *POC:default config* state. This initialization runs for 4.8 µs and will delay the state transition from *POC:default config* into *POC:config*.

51.7.3 Memory error injection out of *POC:default config*

This section provides information for application-driven memory error injection out of *POC:default config*. The CC provides means to inject memory errors from the application without any impacts to the internal protocol operation of the CC.

51.7.3.1 CHI LRAM error injection out of *POC:default config*

The CC will never perform any internal read access from the *LRAM ECC Error Test Registers (FR_LEETRn)*. Any memory errors injected into these CHI LRAM locations will never be detected by internal access, independent from the protocol state.

The application should use these registers and related CHI LRAM location to inject memory errors into the CHI LRAM. The injection sequence is described in [Section 51.6.25.1, “CHI LRAM error injection”](#).

51.7.3.2 PE DRAM error injection out of *POC:default config*

The CC will never perform any internal read access from the PE DRAM address 0x70. This is the only PE DRAM address writable by the application out of the *POC:default config* state.

The application should use these PE DRAM locations to inject memory errors into the PE DRAM. The injection sequence is described in [Section 51.6.25.2, “PE DRAM error injection”](#).

51.7.4 Shutdown sequence

This section describes a secure shutdown sequence to stop the CC gracefully. The main targets of this sequence are

- Finish all ongoing reception and transmission
- Do not corrupt FlexRay bus and do not disturb ongoing FlexRay bus communication

For a graceful shutdown, the application should perform the following tasks:

1. Disable all enabled message buffers.
 - a) Repeatedly write 1 to FR_MBCCSRn[EDT] until FR_MBCCSRn[EDS] == 0.
2. Stop the Protocol Engine.
 - a) Issue HALT command via [Protocol Operation Control Register \(FR_POCR\)](#).
 - b) Wait for *POC:halt* in [Protocol Status Register 0 \(FR_PSR0\)](#).

51.7.5 Number of usable message buffers

This section describes the required minimum CHI clock frequency for a specified number of utilized message buffers configured in the [Message Buffer Segment Size and Utilization Register \(FR_MBSSUTR\)](#), a configured mini-slot length [gdMinislot](#), and a configured nominal macrotick length [gdMacrotick](#)^(t).

Additional constraints for the minimum CHI clock frequency are given in [Section 51.3, "Controller Host Interface clocking"](#).

The CC uses a sequential search algorithm to determine the individual message buffer assigned or subscribed to the next slot. This search is started at the start of the slot and must be finished before the start of the next slot.

The shortest FlexRay slot is a corrected empty dynamic slot. A corrected empty dynamic slot is a mini-slot and consists of [gdMinislot](#) corrected macroticks with a duration of [gdMacrotick](#). The minimum duration of an corrected macrotick is [gdMacrotick](#)_{min} = 39 μT. This results in a minimum length of an correct slot

Equation 69 $\Delta_{\text{slotmin}} = 39 \cdot \text{pdMicrotick} \cdot \text{gdMinislot}$

The message buffer search engine runs on the CHI clock and evaluates one individual message buffer per CHI clock cycle. For internal status update operations and to account for clock domain crossing jitter, an additional amount of 27 CHI clock cycles is required to ensure correct search engine operation.

For a given number of utilized message buffers [FR_MBSSUTR\[LAST_MB_UTIL\] + 1](#) and for a given CHI clock frequency f_{chi} , this results in a search duration of

Equation 70 $\Delta_{\text{search}} = \frac{1}{f_{\text{chi}}} \cdot (\text{FR_MBSSUTR[LAST_MB_UTIL]} + 27)$

The message buffer search must be finished within one slot which requires that [Equation 71](#) must be fulfilled:

Equation 71 $\Delta_{\text{search}} \leq \Delta_{\text{slotmin}}$

This results in the formula given in [Equation 72](#) which determines the required minimum CHI frequency for a given number of message buffers that are utilized.

Equation 72 $f_{\text{chi}} \geq \frac{(\text{FR_MBSSUTR[LAST_MB_UTIL]} + 27)}{39 \cdot \text{pdMicrotick} \cdot \text{gdMinislot}}$

The required minimum CHI Clock frequency for a selected set of relevant protocol parameters and for the LAST_MB_UTIL field in the [Message Buffer Segment Size and Utilization Register \(FR_MBSSUTR\)](#) set to 127 is given in [Table 1104](#).

t. see [Section 51.3, "Controller Host Interface clocking"](#) for all constraints of minimum CHI clock frequency.

Table 1104. Minimum f_{chi} [MHz] examples (128 message buffers used)

<i>pdMicrotick</i> [ns]	gdMinislot					
	2	3	4	5	6	7
25.0	79.5	53	39.8	31.8	26.5	22.8
50.0	39.8	26.5	19.9	15.9	13.3	11.4

Note: If the minimum CHI frequency is not met the *CHIERFR[MBS_EF]* flag is set. Refer to [Section 51.5.2.17, "CHI Error Flag Register \(FR_CHIERFR\)"](#) for details.

51.7.6 Protocol Control Command Execution

This section considers the issues of the protocol control command execution.

The application issues any of the protocol control commands listed in the POCCMD field of [Table 984](#) by writing the command to the POCCMD field of the [Protocol Operation Control Register \(FR_POCCR\)](#). As a result the CC sets the BSY bit while the command is transferred to the PE. When the PE has accepted the command, the BSY flag is cleared. All commands are accepted by the PE.

The PE maintains a protocol command vector. For each command that was accepted by the PE, the PE sets the corresponding command bit in the protocol command vector. If a command is issued while the corresponding command bit is set, the command is not queued and is lost.

If the command execution block of the PE is idle, it selects the next accepted protocol command with the highest priority from the current protocol command vector according to the protocol control command priorities given in [Table 1105](#). If the current protocol state does not allow the execution of this protocol command (see POC state changes in *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*) the CC asserts the illegal protocol command interrupt flag IPC_IF in the [Protocol Interrupt Flag Register 1 \(FR_PIFR1\)](#). The protocol command is not executed in this case.

Some protocol commands may be interrupted by other commands or the detection of a fatal protocol error as indicated by [Table 1105](#). If the application issues the FREEZE or READY command, or if the PE detects a fatal protocol error, some commands already stored in the command vector will be removed from this vector.

Table 1105. Protocol control command priorities

Protocol command	Priority	Interrupted by	Cleared and terminated by
FREEZE	(highest) 1	none	
READY	2		
CONFIG_COMPLETE	3		

Table 1105. Protocol control command priorities (continued)

Protocol command	Priority	Interrupted by	Cleared and terminated by
ALL_SLOTS	4	FREEZE, READY, CONFIG_COMPLETE, fatal protocol error	FREEZE, READY, CONFIG_COMPLETE, fatal protocol error
ALLOW_COLDSTART	5		
RUN	6		FREEZE, fatal protocol error
WAKEUP	7		FREEZE, fatal protocol error
DEFAULT_CONFIG	8		FREEZE, fatal protocol error
CONFIG	9		
HALT	(lowest) 10		FREEZE, READY, CONFIG_COMPLETE, fatal protocol error

51.7.7 Message buffer search on simple message buffer configuration

This section describes the message buffer search behavior for a simplified message buffer configuration. The FIFO behavior is not considered in this section.

51.7.7.1 Simple message buffer configuration

A simple message buffer configuration is a configuration that has at most one transmit message buffer and at most one receive message buffer assigned to a slot S . The simple configuration used in this section utilizes two message buffers, one single buffered transmit message buffer and one receive message buffer.

The transmit message buffer has the message buffer number t and has following configuration

Table 1106. Transmit buffer configuration

Register	Field	Value	Description
FR_MBCCSR t	MTD	1	Transmit buffer
FR_MBCCFR t	MTM	0	Event Transition mode
	CHA	1	Assigned to channel A
	CHB	0	Not assigned to channel B
	CCFE	1	Cycle counter filter enabled
	CCFMSK	000011	Cycle set = $\{4n\} = \{0,4,8,12,\dots\}$
	CCFVAL	000000	
FR_MBFIDR t	FID	S	Assigned to slot S

The availability of data in the transmit buffer is indicated by the commit bit FR_MBCCSR t [CMT] and the lock bit FR_MBCCSR t [LCKS].

The receive message buffer has the message buffer number r and has following configuration

Table 1107. Receive buffer configuration

Register	Field	Value	Description
FR_MBCCSR r	MTD	0	Receive buffer
FR_MBCCFR r	MTM	—	n/a
	CHA	1	Assigned to channel A
	CHB	0	Not assigned to channel B
	CCFE	1	Cycle counter filter enabled
	CCFMSK	000001	Cycle set = $\{2n\} = \{0,2,4,6,\dots\}$
	CCFVAL	000000	
FR_MBFIDR r	FID	S	Subscribed slot

Furthermore the assumption is that both message buffers are enabled (FR_MBCCSR t [EDS] = 1 and FR_MBCCSR r [EDS] = 1)

Note: The cycle set $\{4n+2\} = \{2,6,10,\dots\}$ is assigned to the receive buffer only.
The cycle set $\{4n\} = \{0,4,8,12,\dots\}$ is assigned to both buffers.

51.7.7.2 Behavior in static segment

In this case, both message buffers are assigned to a slot S in the *static* segment.

The configuration of a transmit buffer for a static slot S assigns this slot to the node as a transmit slot. The FlexRay protocol requires:

- When a slot occurs, if the slot is assigned to a node on a channel that node must transmit either a normal frame or a null frame on that channel. Specifically, a null frame will be sent if there is no data ready, or if there is no match on a transmit filter (cycle counter filtering, for example).

Regardless of the availability of data and the cycle counter filter, the node will transmit a frame in the static slot S . In any case, the result of the message buffer search will be the transmit message buffer t . The receive message buffer r will not be found, no reception is possible.

51.7.7.3 Behavior in dynamic segment

In this case, both message buffers are assigned to a slot s in the *dynamic* segment. The FlexRay protocol requires:

- When a slot occurs, if a slot is assigned to a node on a channel that node only transmits a frame on that channel if there is data ready and there is a match on relevant transmit filters (no null frames are sent).

The transmission of a frame in the dynamic segment is determined by the availability of data and the match of the cycle counter filter of the transmit message buffer.

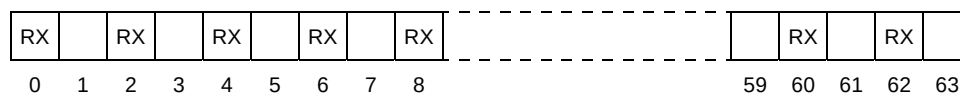
51.7.7.3.1 Transmit Data Not Available

If transmit data are *not available*, that is, the transmit buffer is not committed FR_MBCCSR{[CMT]=0 and/or locked FR_MBCCSR{[LCKS]=1,

- For the cycles in the set $\{4n\}$, which is assigned to both buffers, the receive buffer will be found and the node can receive data, and
- For the cycles in the set $\{4n+2\}$, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive cycles are shown in [Figure 1103](#).

Figure 1103. Transmit data not available



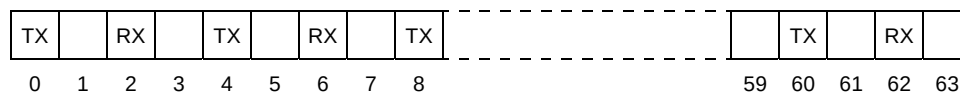
51.7.7.3.2 Transmit data available

If transmit data are *available*, that is, the transmit buffer is committed FR_MBCCSRt[CMT] = 1 and not locked FR_MBCCSRt[LCKS] = 0,

- For the cycles in the set $\{4n\}$, which is assigned to both buffers, the transmit buffer will be found and the node transmits data.
- For the cycles in the set $\{4n+2\}$, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive and transmit cycles are shown in [Figure 1104](#).

Figure 1104. Transmit data available



52 Inter-Integrated Circuit (I²C)

52.1 Overview

This chapter describes the Inter-Integrated Circuit (I²C) bus module implemented on this chip and presents the following topics:

- [Section 52.2, Introduction to I²C](#)
- [Section 52.3, External signal descriptions](#)
- [Section 52.4, Memory map and register definition](#)
- [Section 52.5, Functional description](#)
- [Section 52.6, Initialization/application information](#)

52.2 Introduction to I²C

This section presents the following topics:

- [Section 52.2.1, Definition: I²C module](#)
- [Section 52.2.2, Advantages of the I²C bus](#)
- [Section 52.2.3, Module block diagram](#)
- [Section 52.2.4, Features](#)
- [Section 52.2.5, Modes of operation](#)
- [Section 52.2.6, Definition: I²C conditions](#)

52.2.1 Definition: I²C module

The I²C module is a functional unit that provides a two-wire — serial data (SDA) and serial clock (SCL) — bidirectional serial bus that provides a simple and efficient method of data exchange between this chip and other devices, such as microcontrollers, EEPROMs, real-time clock devices, analog-to-digital converters, and LCDs.

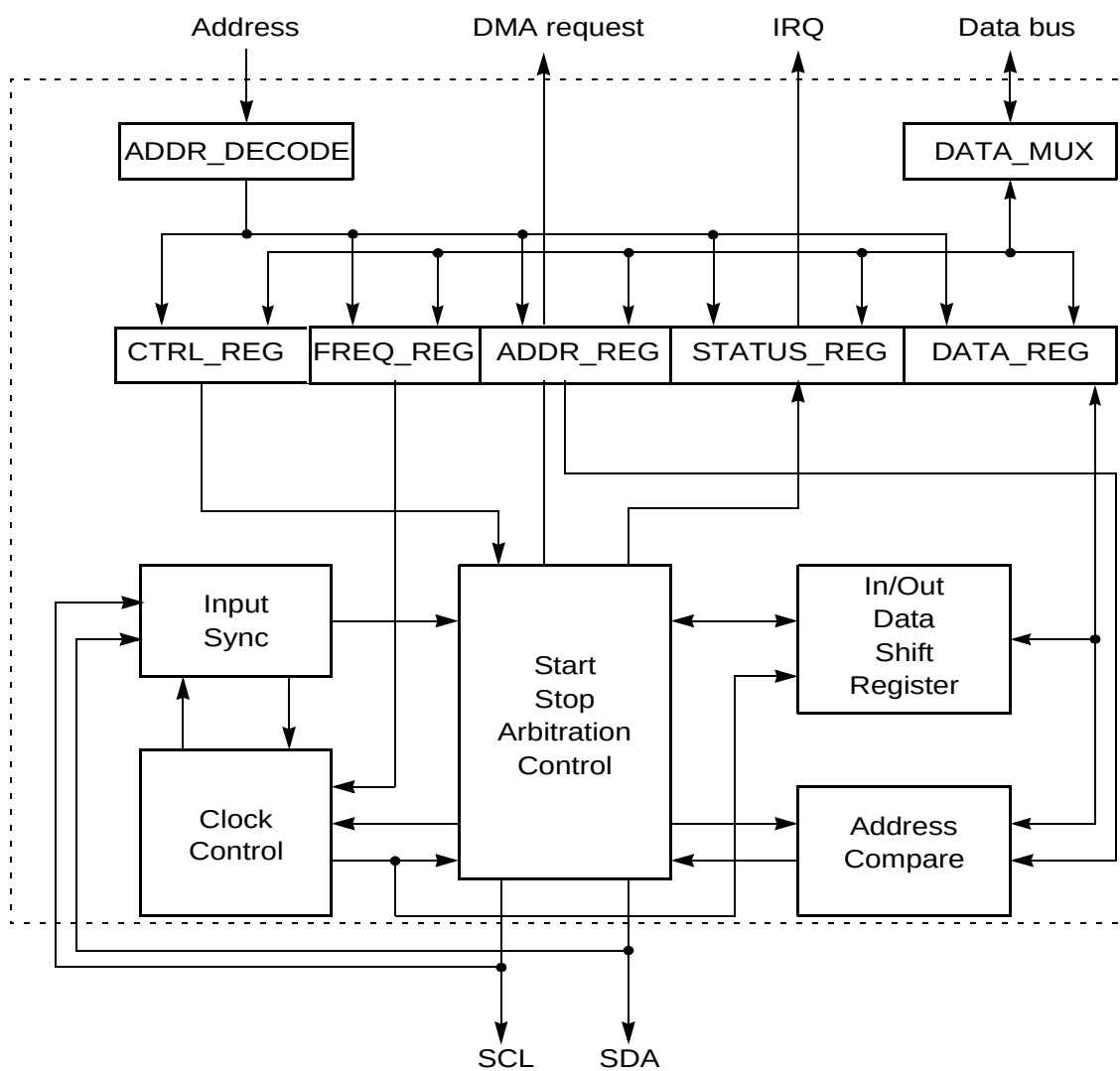
52.2.2 Advantages of the I²C bus

The synchronous, multiple-master two-wire I²C bus:

- Minimizes interconnections between devices
- Allows the connection of additional devices to the bus for expansion and system development
- Includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously
- Does not require an external address decoder

52.2.3 Module block diagram

[Figure 1105](#) shows a block diagram of the I²C module.

Figure 1105. I²C block diagram

52.2.4 Features

The I²C module has the following key features:

- Compatible with I²C bus standard^(u)
- Operating speeds
 - Up to 100 kbps in Standard Mode
 - Up to 400 kbps in Fast Mode
 - Operation at higher baud rates (up to a maximum of module clock/20) with reduced bus loading
 - Actual baud rate dependent on the SCL rise time (which depends on external pull-up resistor values and bus loading)
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Basic DMA interface
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF

52.2.5 Modes of operation

The I²C module supports the chip modes described in [Table 1108](#).

Table 1108. Chip modes supported by the I²C module

Chip mode	Description	Important notes
RUN	Basic mode of operation	—
STOP	The lowest-power mode that allows the chip to turn off all the clocks to the I ² C module	The I ² C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus.
IPG DEBUG	Allows the chip to freeze all ongoing activities (such as an ongoing transaction, counter values, and register status) for debugging	See Section 52.5.6, IPG DEBUG mode .

In addition to chip modes, the I²C module has several module-specific modes. These are described in [Table 1109](#).

u. Compliant with I²C 2.0 standard with the exception that HS (high speed) mode is not supported

Table 1109. Module-specific modes supported by the I²C module

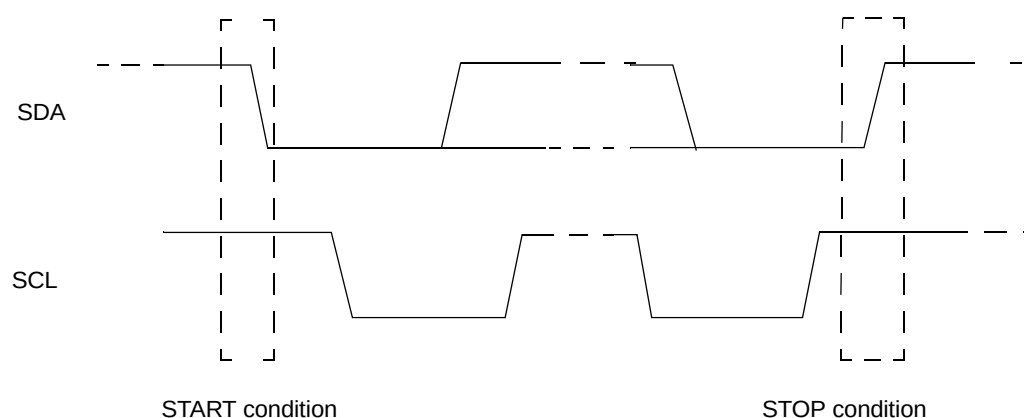
Module mode	Description	Important notes
Master mode	The I ² C module is the driver of the SDA line.	- Do not use the I²C module's slave address as a calling address. - The I²C module cannot be a master and a slave simultaneously.
Slave mode	The I ² C module is not the driver of the SDA line.	- Enable the I²C module before a START condition from a non-I²C master is detected. - By default the I²C module performs as a slave receiver.

52.2.6 Definition: I²C conditions

Table 1110 shows the I²C-specific conditions defined for the I²C module.

Table 1110. I²C Conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data. It is defined as a high-to-low transition of SDA while SCL is high, as shown in Figure 1106 .
STOP	A condition generated by the master to terminate a transfer and free the bus. It is defined as a low-to-high transition of SDA while SCL is high, as shown in Figure 1106 .
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.

**Figure 1106. START and STOP conditions**

52.3 External signal descriptions

This section presents the following topics:

- [Section 52.3.1, Signal overview](#)
- [Section 52.3.2, Detailed external signal descriptions](#)

52.3.1 Signal overview

The I²C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The logical AND function is performed on both signals with external pull-up resistors. For the electrical characteristics of these signals, see the data sheet for this chip.

52.3.2 Detailed external signal descriptions

The SDA and SCL signals are described in [Table 1111](#).

Table 1111. External signal descriptions

Signal	Description
SCL	Bidirectional serial clock line of the module, compatible with the I ² C bus specification
SDA	Bidirectional serial data line of the module, compatible with the I ² C bus specification

52.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the I²C module. It presents the following topics:

- [Section 52.4.1, I²C memory map](#)
- [Section 52.4.2, Register accessibility](#)
- [Section 52.4.3, Register figure conventions](#)
- [Section 52.4.4, I²C Bus Address Register \(IBAD\)](#)
- [Section 52.4.5, I²C Bus Frequency Divider Register \(IBFD\)](#)
- [Section 52.4.6, I²C Bus Control Register \(IBCR\)](#)
- [Section 52.4.7, I²C Bus Status Register \(IBSR\)](#)
- [Section 52.4.8, I²C Bus Data I/O Register \(IBDR\)](#)
- [Section 52.4.9, I²C Bus Interrupt Config Register \(IBIC\)](#)
- [Section 52.4.10, I²C Bus Debug Register \(IBDBG\)](#)

52.4.1 I²C memory map

The memory map for the I²C module is given below in [Table 1112](#). The total address for each register is the sum of the base address for the I²C module and the address offset for each register.

Table 1112. I²C memory map

Address Offset	Register	Access	Reset value	Section/Page
0x0000	IBAD—I ² C Bus Address Register	R/W	0x00	52.4.4/2071
0x0001	IBFD—I ² C Bus Frequency Divider Register	R/W	0x00	52.4.5/2071
0x0002	IBCR—I ² C Bus Control Register	R/W	0x80	52.4.6/2072
0x0003	IBSR—I ² C Bus Status Register	R/W	0x80	52.4.7/2073
0x0004	IBDR—I ² C Bus Data I/O Register	R/W	0x00	52.4.8/2075
0x0005	IBIC—I ² C Bus Interrupt Config Register	R/W	0x00	52.4.9/2076
0x0006	IBDBG—I ² C Bus Debug Register	R/W	0x00	52.4.10/2076
0x0007 – 0x3FFF	Reserved ⁽¹⁾			

1. If enabled at the SoC level, reads or writes to these registers will cause bus aborts. Refer to the System Services Module documentation for more details.

52.4.2 Register accessibility

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries.

As an example, the IBFD is at address offset 0x01, and can be accessed in any of the following ways:

- 8-bit R/W access of address offset 0x0001
- Second byte of 16-bit R/W access of address offset 0x0000
- Second byte of 32-bit R/W access of address offset 0x0000

The IBFD register cannot be accessed by a 16- or 32-bit RW operation at address offset 0x0001 because those operations require an address aligned to a 16- or 32-bit boundary.

52.4.3 Register figure conventions

The register figures show the field structure using the conventions in [Figure 1107](#).

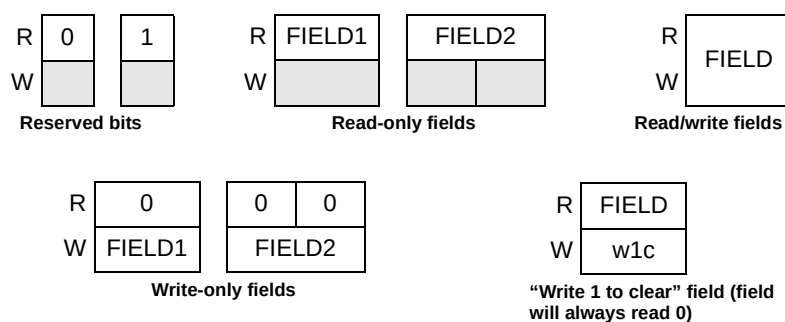


Figure 1107. Register figure conventions

52.4.4 I²C Bus Address Register (IBAD)

This register contains the address the I²C Bus will respond to when addressed as a slave. This is not the address sent on the bus during the address transfer.

Figure 1108. I²C Bus Address register (IBAD)

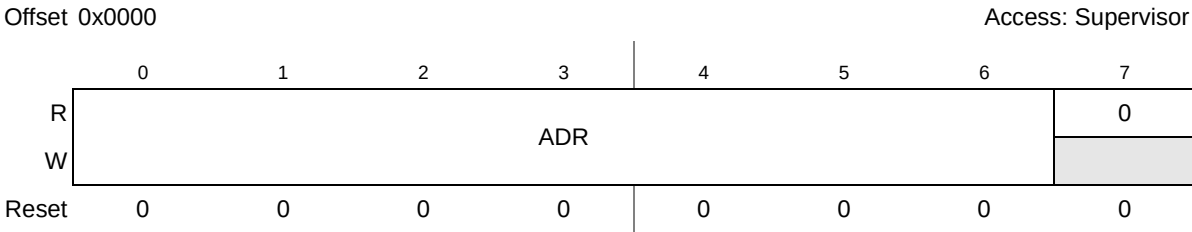


Table 1113. IBAD field descriptions

Field	Description
0–6 ADR	Slave Address. Specific slave address to be used by the I ² C Bus module. <i>Note: The default mode of I²C Bus is slave mode for an address match on the bus.</i>
7	Reserved, should be cleared; will always read 0.

52.4.5 I²C Bus Frequency Divider Register (IBFD)

Figure 1109. I²C Bus Frequency Divider Register (IBFD)

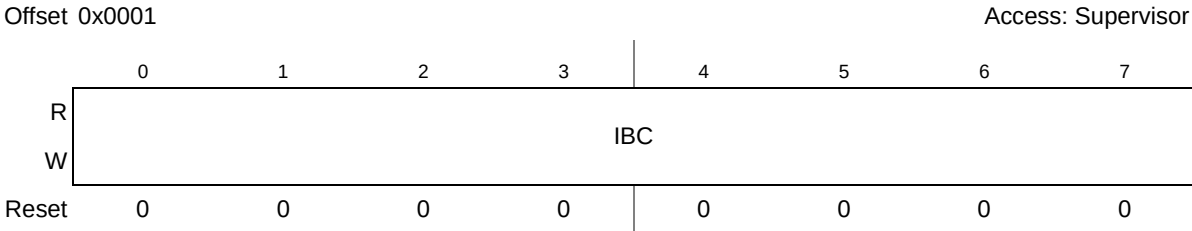


Table 1114. IBFD field descriptions

Field	Description
0–7 IBC	I-Bus Clock Rate. This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider. The IBC bits are decoded to give the Tap and Prescale values as follows: 0–1 Selects the prescaled shift register (see Table 1123) 2–4 Selects the prescaler divider (see Table 1124) 5–7 Selects the shift register tap point (see Table 1125)

52.4.6 I²C Bus Control Register (IBCR)Figure 1110. I²C Bus Control Register (IBCR)

Offset 0x0002

Access: Supervisor

	0	1	2	3	4	5	6	7
R	MDIS	IBIE	MSSL	TXRX	NOACK	0	DMAEN	0
W						RSTA		
Reset	1	0	0	0	0	0	0	0

Table 1115. IBCR field descriptions

Field	Description
0 MDIS	Module disable. This bit controls the software reset of the entire I²C Bus module. 0 The I²C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed. Status register bits (IBSR) are not valid when module is disabled. <i>Note: If the I²C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I²C Bus module losing arbitration, after which, bus operation would return to normal.</i>
1 IBIE	I-Bus Interrupt Enable. 0 Interrupts from the I²C Bus module are disabled. This does not clear any currently pending interrupt condition. 1 Interrupts from the I²C Bus module are enabled. An I²C Bus interrupt occurs provided the IBIF bit in the status register is also set.
2 MSSL	Master/Slave mode select. When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. This field is cleared without generating a STOP signal when the master loses arbitration. 0 Slave Mode 1 Master Mode
3 TXRX	Transmit/Receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive 1 Transmit

Table 1115. IBCR field descriptions (continued)

Field	Description
4 NOACK	Data Acknowledge disable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I ² C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. <i>Note: Values written to this bit are only used when the I²C Bus is a receiver, not a transmitter.</i> 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)
5 RSTA	Repeat Start. Writing a one to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 0 No effect 1 Generate repeat start cycle
6 DMAEN	DMA Enable. When this bit is set, the DMA Tx and Rx lines will be asserted when the I ² C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the I ² C module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. See the DMA Application Information section for more details. 0 Disable the DMA TX/RX request signals 1 Enable the DMA TX/RX request signals

52.4.7 I²C Bus Status Register (IBSR)

Figure 1111. I²C Bus Status Register (IBSR)

Offset 0x0003

Access: Supervisor⁽¹⁾

	0	1	2	3	4	5	6	7
R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
W				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

1. With the exception of IBIF and IBAL are w1c.

Table 1116. IBSR field descriptions

Field	Description
0 TCF	Transfer complete. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. <i>Note: This bit is only valid during or immediately following a transfer to the I²C module or from the I²C module.</i> 0 Transfer in progress 1 Transfer complete
1 IAAS	Addressed as a slave. When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set the TXRX field accordingly. Writing to the I-Bus Control Register clears this bit. 0 Not addressed 1 Addressed as a slave
2 IBB	Bus busy. This bit indicates the status of the bus. When a START signal is detected, IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. 0 Bus is Idle 1 Bus is busy
3 IBAL	Arbitration Lost. The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: – SDA is sampled low when the master drives a high during an address or data transmit cycle. – SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. – A start cycle is attempted when the bus is busy. – A repeated start cycle is requested in slave mode. – A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
4 Reserved	Reserved
5 SRW	Slave Read/Write. When the IAAS bit is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By programming this bit, the CPU can select slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave

Table 1116. IBSR field descriptions (continued)

Field	Description
6 IBIF	I-Bus Interrupt Flag. The IBIF bit is set when one of the following conditions occurs: – Arbitration lost (IBAL bit set) – Byte transfer complete (TCF bit set and DMAEN bit not set) – Addressed as slave (IAAS bit set) – NoAck from Slave (MS & Tx bits set) – I²C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. In DMA mode (DMAEN set) a byte transfer complete condition will not trigger the setting of IBIF. All other conditions still apply.
7 RXAK	Received Acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete. 0 Acknowledge received 1 No acknowledge received

52.4.8 I²C Bus Data I/O Register (IBDR)

In master transmit mode, when data is written to the IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.

Note: The IBCR[TXRX] field must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I²C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the most recent byte received while the I²C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I²C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MSSL is used for the address transfer and should comprise the calling address (in position DATA[7:1]) concatenated with the required R/W bit (in position D0).

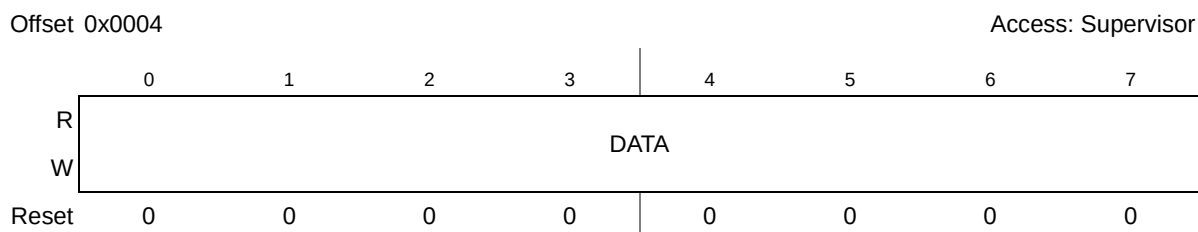
Figure 1112. I²C Bus Data I/O Register (IBDR)

Table 1117. IBDR field descriptions

Field	Description
0–7 DATA	Data transmitted or received

52.4.9 I²C Bus Interrupt Config Register (IBIC)

Figure 1113. I²C Bus Interrupt Config Register (IBIC)

Offset 0x0005

Access: Supervisor

	0	1	2	3	4	5	6	7
R	BIIE	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Table 1118. IBIC field descriptions

Field	Description
0 BIIE	Bus Idle Interrupt Enable bit. This config bit can be used to enable the generation of an interrupt once the I ² C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I ² C bus. 0 Bus Idle Interrupts disabled 1 Bus Idle Interrupts enabled
1–7 Reserved	Reserved

52.4.10 I²C Bus Debug Register (IBDBG)

Figure 1114. I²C Bus Debug Register (IBDBG)

Offset 0x0006

Access: Supervisor

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	IPG_DEBUG_HALTED	IPG_DEBUG_EN
W								
Reset	0	0	0	0	0	0	0	0

Table 1119. IBDBG field descriptions

Field	Description
6 IPG_DEBUG_HALTED	Debug Halted Bit: This is a status bit which can be read back after asserting the debug enable signal so as to know if the IP has entered the debug mode or not. 0 IP is still executing a transaction 1 IP has entered the debug mode
7 IPG_DEBUG_EN	Debug enable bit. This bit is used to enable IP enter the debug mode provided the IPG DEBUG signal is high. All the registers, counter values and status bits are frozen and can be accessed by the CPU. 0 Normal operation, Bus Idle Interrupts disabled 1 IP is in debug mode <i>Note: If the assertion of this bit along with the IPG DEBUG signal happens in the middle of a transaction, IP enters the debug mode only after successful completion of the current transaction after which no further transaction can take place until the debug mode is exited.</i>

52.5 Functional description

This section presents the following topics:

- [Section 52.5.1, Notes about module operation](#)
- [Section 52.5.2, Transactions](#)
- [Section 52.5.3, Arbitration procedure](#)
- [Section 52.5.4, Clock behavior](#)
- [Section 52.5.5, Interrupts](#)
- [Section 52.5.6, IPG DEBUG mode](#)
- [Section 52.5.7, DMA interface](#)

52.5.1 Notes about module operation

- The I²C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I²C module is acting as a master, it must not try to call its own slave address.

52.5.2 Transactions

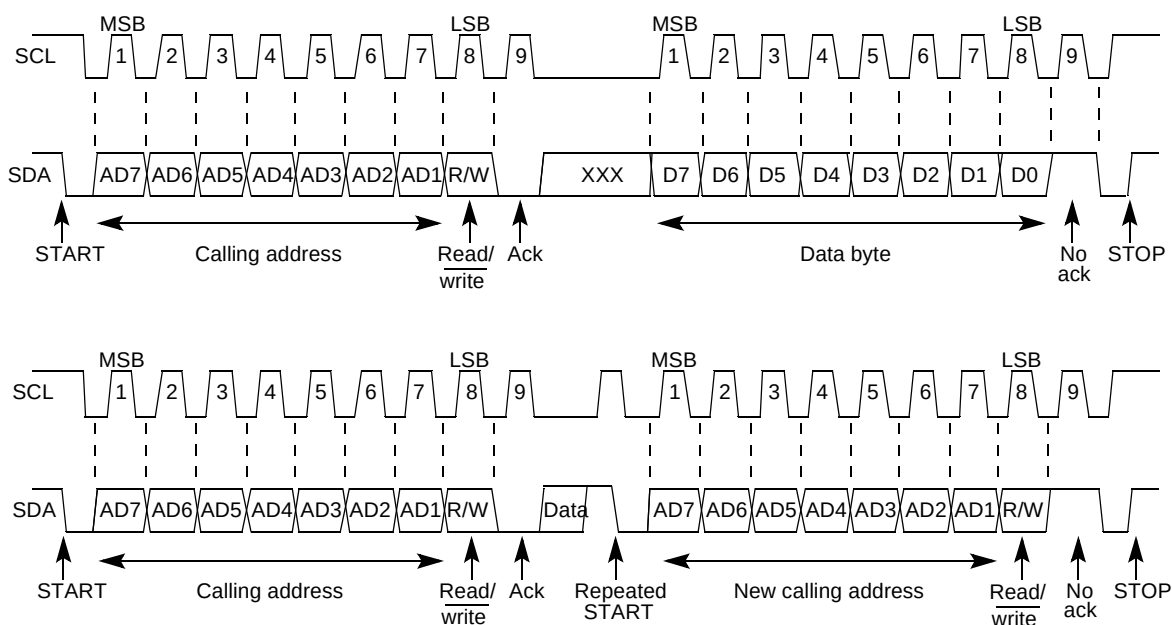
This section presents the following topics:

- [Section 52.5.2.1, Protocol overview](#)
- [Section 52.5.2.2, Transaction protocol definitions](#)
- [Section 52.5.2.4, High-level protocol steps](#)
- [Section 52.5.2.5, START condition](#)
- [Section 52.5.2.6, Slave address transmission](#)
- [Section 52.5.2.7, Data transmission](#)
- [Section 52.5.2.8, STOP condition](#)
- [Section 52.5.2.9, Repeated START condition](#)

52.5.2.1 Protocol overview

Figure 1115 shows the behavior of SCL and SDA during a typical I²C transaction.

Figure 1115. I²C transaction protocol



52.5.2.2 Transaction protocol definitions

This section defines several important terms presented in Figure 1115.

Table 1120. I²C definitions

Term	Definition
START	A START condition, as defined in Section 52.2.6, Definition: I2C conditions
STOP	A STOP condition, as defined in Section 52.2.6, Definition: I2C conditions
Calling (slave) address	A seven-bit address used to identify a slave on the I ² C bus. The requirements for specifying this address are presented in Section 52.5.2.3, I2C calling address requirements .
Read/write (R/W)	A bit that specifies the direction of the data transfer to the slave as follows: – 0 = The data is being transferred from the master to the slave ("write") – 1 = The data is being transferred from the slave to the master ("read")
Ack	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

52.5.2.3 I²C calling address requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

52.5.2.4 High-level protocol steps

The I²C protocol conceptually supports two types of transfers, which are illustrated in [Figure 1115](#). The significant steps in these transfers are presented in [Table 1121](#). Details of each of these steps are presented in subsequent sections.

Table 1121. I²C high-level protocol steps

Standard Transfer	Repeated START Transfer
1. START condition 2. Slave target or general call address transmission 3. Acknowledgment from slave 4. Data transfer 5. STOP condition 6. (repeat Steps 1–4)	1. START condition 2. Slave target or general call address transmission 3. Acknowledgment from slave 4. Data transfer 5. Repeated START condition 6. (repeat Steps 2–4 as needed) 7. STOP condition. 8. (repeat Steps 1–7)

52.5.2.5 START condition

When the bus is free, that is, no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START condition (see [Section 52.2.6, Definition: I²C conditions](#)). This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

52.5.2.6 Slave address transmission

The master transmits the slave address immediately after the START condition (see [Section 52.5.2.5, START condition](#)). The process of slave address transmission is presented in [Table 1122](#).

Table 1122. Slave address transmission process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/W bit.
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: – The acknowledge bit is set: The master must initiate a data transfer followed by either a STOP condition or a repeated START condition. – The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

52.5.2.7 Data transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data
- Awakens all slaves
- Proceeds on a byte-by-byte basis in the direction specified by the $\overline{R/W}$ bit sent by the calling master

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.
- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.
- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

52.5.2.8 STOP condition

The master can terminate the communication by generating a STOP condition (see [Section 52.2.6, Definition: I2C conditions](#)). It can do so even if the slave has generated an acknowledge, at which point the slave must release the bus.

A master is not required to send a STOP condition at the end of every transfer. For more information, see [Section 52.5.2.9, Repeated START condition](#).

52.5.2.9 Repeated START condition

The I²C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of [Figure 1115](#).

52.5.3 Arbitration procedure

The I²C bus is a true multi-master bus that allows more than one master to be connected to it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic "1" while another master transmits logic "0". The losing masters immediately switch over to slave receive mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

52.5.4 Clock behavior

This section presents the following topics:

- [Section 52.5.4.1, Clock synchronization](#)
- [Section 52.5.4.2, Clock stretching](#)
- [Section 52.5.4.3, Handshaking](#)
- [Section 52.5.4.4, Clock rate and IBFD settings](#)

52.5.4.1 Clock synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see [Figure 1116](#)). When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

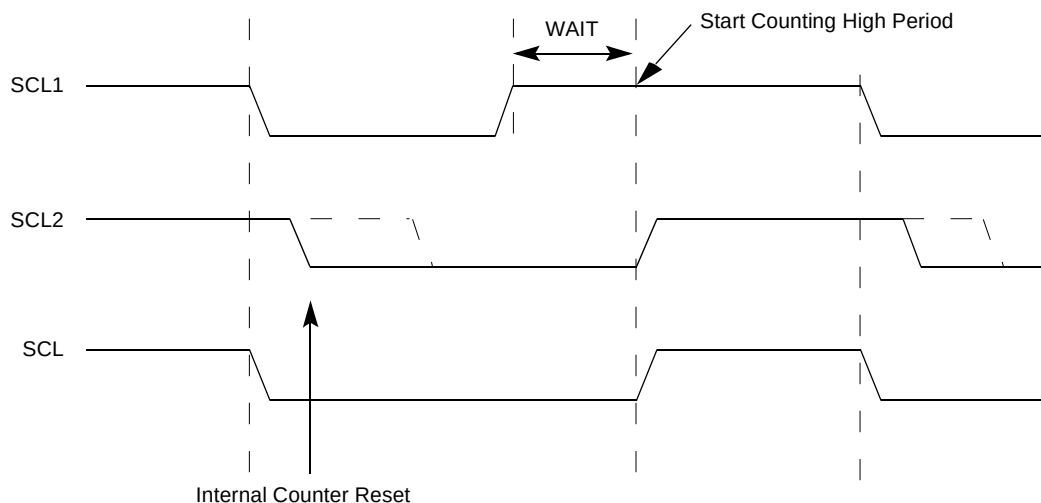


Figure 1116. I²C bus clock synchronization

52.5.4.2 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

52.5.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it

halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

52.5.4.4 Clock rate and IBFD settings

The following tables describe the settings of several fields in the IBFD register (see [Section 52.4.5, I2C Bus Frequency Divider Register \(IBFD\)](#)).

Table 1123. I-Bus multiplier factor

IBC	MUL
00	01
01	02
10	04
11	Reserved

Table 1124. I-Bus prescaler divider values

IBC	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

Table 1125. I-Bus tap and prescale values

IBC	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of [Table 1124](#). All subsequent tap points are separated by 2^{IBC} as shown in the tap2tap column in [Table 1124](#). The SCL Tap is used to generate the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to the change of state of SDA that is the SDA Hold time.

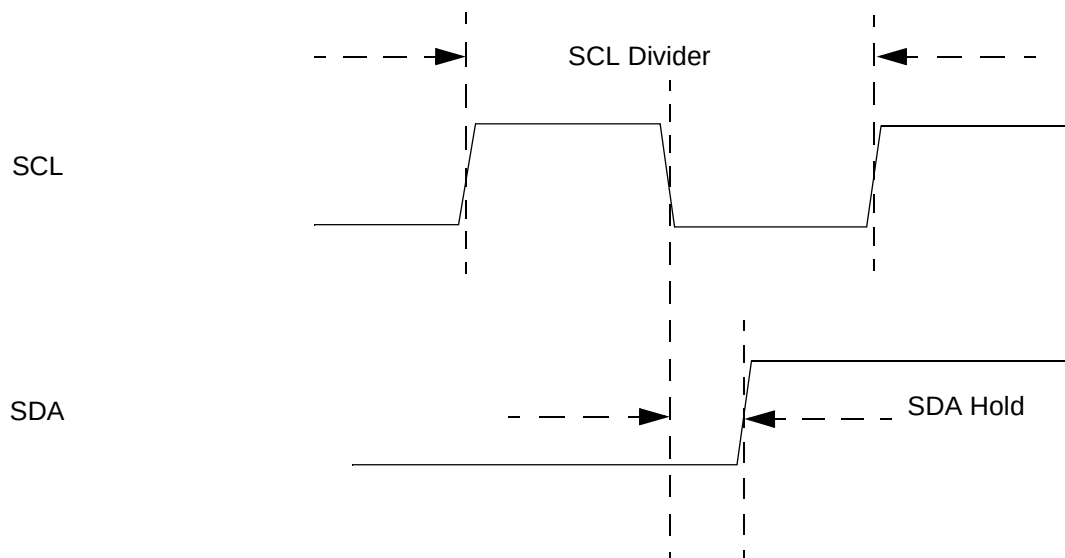


Figure 1117. SDA Hold time

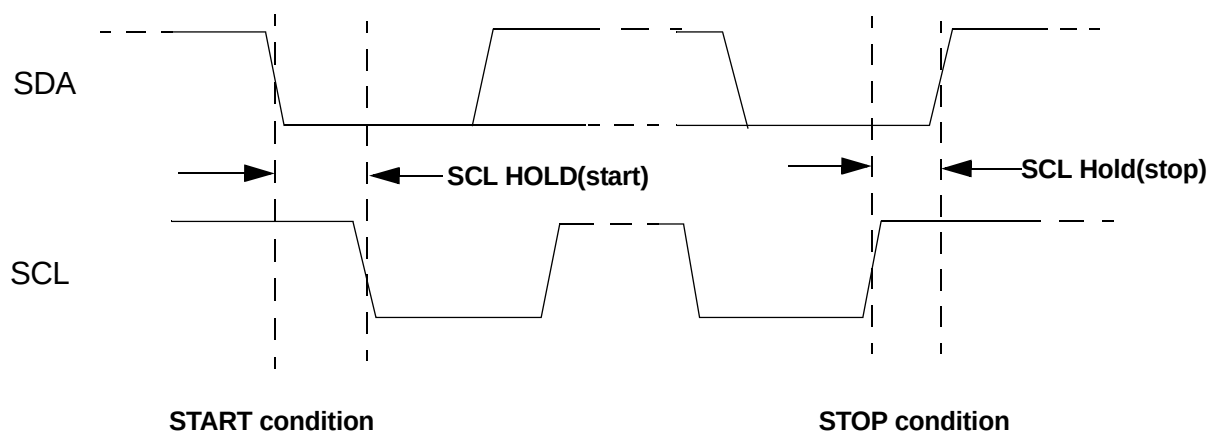


Figure 1118. SCL Divider and SDA Hold

The equation used to generate the divider values from the IBFD bits is:

$$\text{Equation 73} \quad \text{SCL Divider} = \text{MUL} \times \{2 \times (\text{scl2tap} + [(\text{SCL_Tap} - 1) \times \text{tap2tap}] + 2)\}$$

The SDA Hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in [Table 1126](#). The equation used to generate the SDA Hold value from the IBFD bits is:

$$\text{Equation 74} \quad \text{SDA Hold} = \text{MUL} \times \{\text{scl2tap} + [(\text{SDA_Tap} - 1) \times \text{tap2tap}] + 3\}$$

The equations for SCL Hold values to generate the START and STOP conditions from the IBFD bits are:

Equation 75 $\text{SCL Hold(start)} = \text{MUL} \times [\text{scl2start} + (\text{SCL_Tap} - 1) \times \text{tap2tap}]$

Equation 76 $\text{SCL Hold(stop)} = \text{MUL} \times [\text{scl2stop} + (\text{SCL_Tap} - 1) \times \text{tap2tap}]$

Table 1126. I²C divider and hold values

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=1	00	20	7	6	11
	01	22	7	7	12
	02	24	8	8	13
	03	26	8	9	14
	04	28	9	10	15
	05	30	9	11	16
	06	34	10	13	18
	07	40	10	16	21
	08	28	7	10	15
	09	32	7	12	17
	0A	36	9	14	19
	0B	40	9	16	21
	0C	44	11	18	23
	0D	48	11	20	25
	0E	56	13	24	29
	0F	68	13	30	35
	10	48	9	18	25
	11	56	9	22	29
	12	64	13	26	33
	13	72	13	30	37
	14	80	17	34	41
	15	88	17	38	45
	16	104	21	46	53
	17	128	21	58	65
	18	80	9	38	41
	19	96	9	46	49
	1A	112	17	54	57
	1B	128	17	62	65
	1C	144	25	70	73
	1D	160	25	78	81
	1E	192	33	94	97
	1F	240	33	118	121
	20	160	17	78	81
	21	192	17	94	97
	22	224	33	110	113
	23	256	33	126	129

Table 1126. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=1	24	288	49	142	145
	25	320	49	158	161
	26	384	65	190	193
	27	480	65	238	241
	28	320	33	158	161
	29	384	33	190	193
	2A	448	65	222	225
	2B	512	65	254	257
	2C	576	97	286	289
	2D	640	97	318	321
	2E	768	129	382	385
	2F	960	129	478	481
	30	640	65	318	321
	31	768	65	382	385
	32	896	129	446	449
	33	1024	129	510	513
	34	1152	193	574	577
	35	1280	193	638	641
	36	1536	257	766	769
	37	1920	257	958	961
	38	1280	129	638	641
	39	1536	129	766	769
	3A	1792	257	894	897
	3B	2048	257	1022	1025
	3C	2304	385	1150	1153
	3D	2560	385	1278	1281
	3E	3072	513	1534	1537
	3F	3840	513	1918	1921
MUL=2	40	40	14	12	22
	41	44	14	14	24
	42	48	16	16	26
	43	52	16	18	28
	44	56	18	20	30
	45	60	18	22	32
	46	68	20	26	36
	47	80	20	32	42
	48	56	14	20	30
	49	64	14	24	34

Table 1126. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=2	4A	72	18	28	38
	4B	80	18	32	42
	4C	88	22	36	46
	4D	96	22	40	50
	4E	112	26	48	58
	4F	136	26	60	70
	50	96	18	36	50
	51	112	18	44	58
	52	128	26	52	66
	53	144	26	60	74
	54	160	34	68	82
	55	176	34	76	90
	56	208	42	92	106
	57	256	42	116	130
	58	160	18	76	82
	59	192	18	92	98
	5A	224	34	108	114
	5B	256	34	124	130
	5C	288	50	140	146
	5D	320	50	156	162
	5E	384	66	188	194
	5F	480	66	236	242
	60	320	28	156	162
	61	384	28	188	194
	62	448	32	220	226
	63	512	32	252	258
	64	576	36	284	290
	65	640	36	316	322
	66	768	40	380	386
	67	960	40	476	482
	68	640	28	316	322
	69	768	28	380	386
	6A	896	36	444	450
	6B	1024	36	508	514

Table 1126. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=2	6C	1152	44	572	578
	6D	1280	44	636	642
	6E	1536	52	764	770
	6F	1920	52	956	96
	70	1280	36	636	642
	71	1536	36	764	770
	72	1792	52	892	898
	73	2048	52	1020	1026
	74	2304	68	1148	1154
	75	2560	68	1276	1282
	76	3072	84	1532	1538
	77	3840	84	1916	1922
	78	2560	36	1276	1282
	79	3072	36	1532	1538
	7A	3584	68	1788	1794
	7B	4096	68	2044	2050
	7C	4608	100	2300	2306
	7D	5120	100	2556	2562
	7E	6144	132	3068	3074
MUL=4	80	80	28	24	44
	81	88	28	28	48
	82	96	32	32	52
	83	104	32	36	56
	84	112	36	40	60
	85	120	36	44	64
	86	136	40	52	72
	87	160	40	64	84
	88	112	28	40	60
	89	128	28	48	68
	8A	144	36	56	76
	8B	160	36	64	84
	8C	176	44	72	92
	8D	192	44	80	100
	8E	224	52	96	116
	8F	272	52	120	140
	90	192	36	72	100
	91	224	36	88	116
	92	256	52	104	132

Table 1126. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=4	93	288	52	120	148
	94	320	68	136	164
	95	352	68	152	180
	96	416	84	184	212
	97	512	84	232	260
	98	320	36	152	164
	99	384	36	184	196
	9A	448	68	216	228
	9B	512	68	248	260
	9C	576	100	280	292
	9D	640	100	312	324
	9E	768	132	376	388
	9F	960	132	472	484
	A0	640	68	312	324
	A1	768	68	376	388
	A2	896	132	440	452
	A3	1024	132	504	516
	A4	1152	196	568	580
	A5	1280	196	632	644
	A6	1536	260	760	772
	A7	1920	260	952	964
	A8	1280	132	632	644
	A9	1536	132	760	772
	AA	1792	260	888	900
	AB	2048	260	1016	1028
	AC	2304	388	1144	1156
	AD	2560	388	1272	1284
	AE	3072	516	1528	1540
	AF	3840	516	1912	1924
	30	2560	260	1272	1284
	B1	3072	260	1528	1540
	B2	3584	516	1784	1796
	B3	4096	516	2040	2052
	B4	4608	772	2296	2308
	B5	5120	772	2552	2564
	B6	6144	1028	3064	3076
	B7	7680	1028	3832	3844
	B8	5120	516	2552	2564

Table 1126. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=4	B9	6144	516	3064	3076
	BA	7168	1028	3576	3588
	BB	8192	1028	4088	4100
	BC	9216	1540	4600	4612
	BD	10240	1540	5112	5124
	BE	12288	2052	6136	6148
	BF	15360	2052	7672	7684

52.5.5 Interrupts

This section presents the following topics:

- [Section 52.5.5.1, Interrupt vector](#)
- [Section 52.5.5.2, Interrupt description](#)

52.5.5.1 Interrupt vector

The I²C module uses only one interrupt vector.

Table 1127. Interrupt summary

Interrupt	Offset	Vector	Priority	Source	Description
I ² C Interrupt	—	—	—	IBAL, TCF, IAAS, IBB bits in IBSR register	When any of IBAL, TCF or IAAS bits is set, an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the de-assertion of IBB can also cause an interrupt, indicating that the bus is idle.

52.5.5.2 Interrupt description

There are five types of internal interrupts in the I²C. The interrupt service routine can determine the interrupt type by reading the Status register.

I²C Interrupt can be generated on the following events:

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set and DMAEN bit not set)
- Address Detect condition (IAAS bit set)
- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I²C interrupt is enabled by the IBCR[IBIE] bit. It must be cleared by writing '1' to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the IBIC[BIIIE] bit.

52.5.6 IPG DEBUG mode

This mode allows CPU to debug the I²C by freezing all the counters, registers and status bits. Once the Debug request is asserted along with IBDBG[IPG_DEBUG_EN] bit, I²C comes to a graceful halt after completing all the ongoing transactions. A Debug halted signal IBDBG[IPG_DEBUG_HALTED] is also asserted to indicate that the debug request has been successfully serviced and is de-asserted when the Debug request is de-asserted.

As soon as the I²C module enters IPG DEBUG mode:

- No transaction can take place.
- All the registers, counters and status bits are frozen. They all can be accessed by the CPU to enable the debugging.

The I²C module enters this mode only after successfully completing the current ongoing transaction. No more transaction can take place until the debug signal is de-asserted after which the I²C module starts functioning normally. There is a status halted signal IBDBG[IPG_DEBUG_HALTED] to indicate to the user that the I²C module has entered the IPG DEBUG mode. See [Figure 1119](#) for more details.

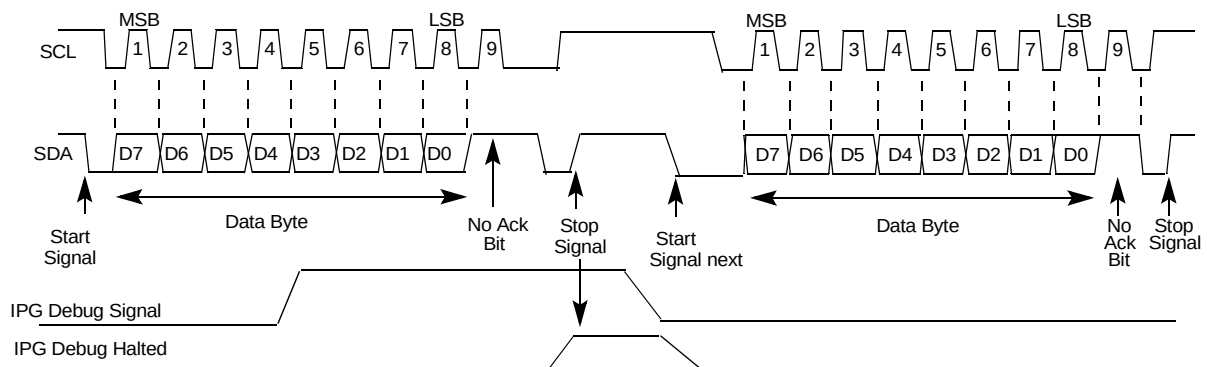


Figure 1119. IPG DEBUG mode

[Figure 1120](#) shows a case of IPG DEBUG mode with repeated START transaction. In this case, if the debug signal is asserted in between the transaction, the entire transaction with multiple repeated start will be completed before the I²C module enters IPG DEBUG mode.

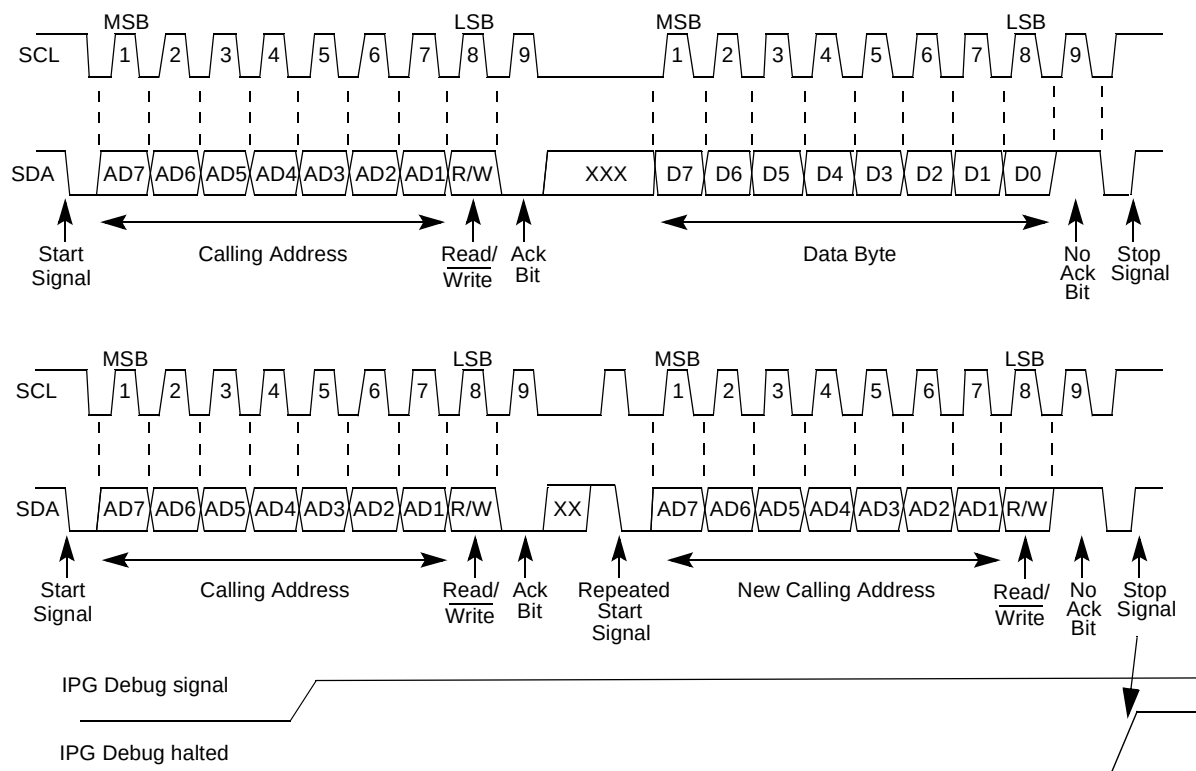


Figure 1120. IPG debug mode with repeated start

52.5.7 DMA interface

A simple DMA interface is implemented so that the I²C can request data transfers with minimal support from the CPU. DMA mode is enabled by setting bit 1 in the Control Register (IBCR).

The DMA interface is operational when the I²C module is configured for Master mode.

At least three bytes of data per frame must be transferred from/to the slave when using DMA mode, although in practice it will only be worthwhile using the DMA mode when there is a large number of data bytes to transfer per frame.

Two internal signals, TX request and RX request, are used to signal the DMA controller when the I²C module requires data to be written or read from the data register.

Further details of the DMA interface can be found in the [Section 52.6, Initialization/application information](#), of this document.

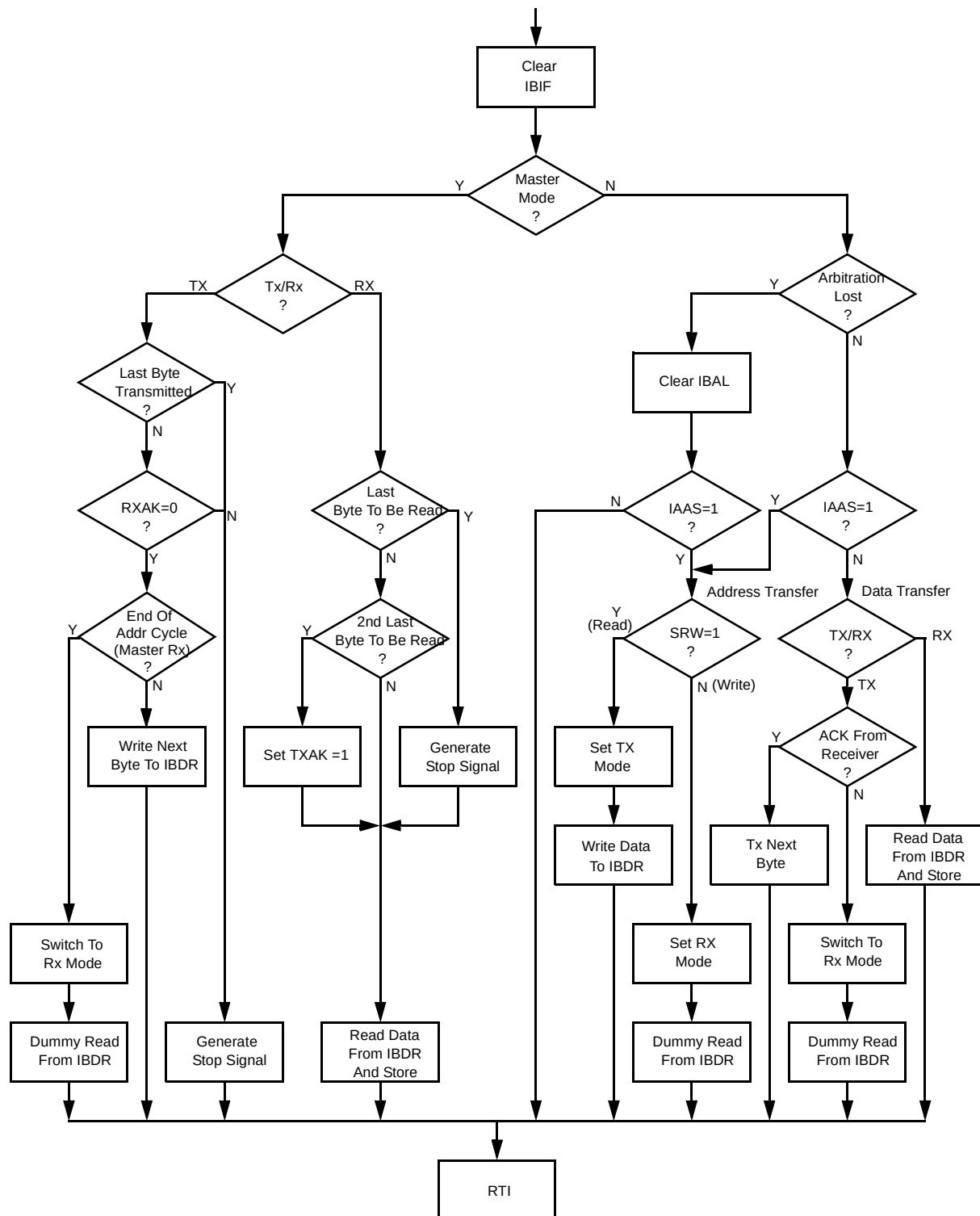
52.6 Initialization/application information

This section presents the following topics:

- [Section 52.6.1, Recommended interrupt service flow](#)
- [Section 52.6.2, General programming guidelines \(for both master and slave mode\)](#)
- [Section 52.6.3, Programming guidelines specific to master mode](#)
- [Section 52.6.4, Programming guidelines specific to slave mode](#)
- [Section 52.6.5, DMA application information](#)

52.6.1 Recommended interrupt service flow

[Figure 1121](#) shows a flowchart for the recommended I²C interrupt service routine. Deviation from the flowchart may result in unpredictable I²C bus behavior.

Figure 1121. Recommended I²C interrupt service routine flowchart

52.6.2 General programming guidelines (for both master and slave mode)

This section provides programming guidelines recommended for the I²C module in both master and slave mode. It presents the following topics:

- [Section 52.6.2.1, Initializing the I²C module](#)
- [Section 52.6.2.2, Software response after a transfer](#)

52.6.2.1 Initializing the I²C module

The following sequence initializes the I²C module:

1. Use the IBFD register to select the required division ratio to obtain SCL frequency from system clock.
2. Use the IBAD register to define the slave address.
3. Clear the IBCR[IBDIS] bit to enable the I²C interface system.
4. Use the IBCR to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not.
5. (Optional) Use the IBIC register to further refine the interrupt behavior.

52.6.2.2 Software response after a transfer

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I²C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing one (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress whenever data register is written to in transmit mode, or during reading out from data register in receive mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I²C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I²C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

When a "Transfer Complete" interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit sent with slave calling address, then the Tx/Rx bit at Master side should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

In slave mode, IAAS bit will get set in IBSR if Slave address (IBAD) matches the Master calling address. This is an indication that Master-Slave data communication can now start. During address cycles (IBSR[IAAS]=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IBSR[IAAS]=0), the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

52.6.3 Programming guidelines specific to master mode

This section presents the following topics:

- [Section 52.6.3.1, Generating START](#)
- [Section 52.6.3.2, Transmit/receive sequence](#)
- [Section 52.6.3.3, Generating STOP](#)
- [Section 52.6.3.4, Generating repeated START](#)
- [Section 52.6.3.5, Loss of arbitration](#)

52.6.3.1 Generating START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I²C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I²C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (IBSR[IBB]==1) // wait in loop for IBB flag to clear
  IBCR[MS/MSL] and IBVR[ ]Tx/Rx] = 1 //master and transmit mode, that is,
  generate start condition
  IBDR = calling_address // send the calling address to the data register
  while (bit 5, IBSR ==0) // wait in loop for IBB flag to be set
```

52.6.3.2 Transmit/receive sequence

Below sequence to be followed in case of Master Transmit (Address/Data):

- Clear IBIF in IBSR register.
- Write data in Data Register (IBDR).
- TCF bit will get cleared in IBSR register when transfer is in progress.
- TCF bit will get set in IBSR register when transfer is complete.
- Wait for IBIF to get set in IBSR register, then read IBSR register to determine its source
 - TCF = 1 that is transfer is complete.
 - No Acknowledge condition (RXAK = 1) is found.
 - IBB = 0 that is Bus has transitioned from Busy to Idle state.
 - If IBB = 1, ignore check of Arbitration Loss (IBAL = 1).
 - Ignore Address Detect (IAAS = 1) for Master mode (valid only for Slave mode).
- Check RXAK in IBSR for an acknowledge from slave.

Below sequence to be followed in case of Slave Receive (Address/Data):

- a) Clear IBIF in IBSR register.
- b) TCF bit will get cleared in IBSR register when transfer is in progress for address transfer.
- c) TCF bit will get set in IBSR register when transfer is complete.
- d) Wait for IBIF to get set in IBSR register. Then read IBSR register to determine its source:
 - Address Detect has occurred (IAAS = 1) — determination of Slave mode.
- e) Clear IBIF.
- f) Wait till TCF bit gets cleared in IBSR register (that is “Transfer under Progress” condition is reached for data transfer).
- g) Wait till TCF bit gets cleared in IBSR register (proof that Transfer Completes from “Transfer under Progress” state).
- h) Wait till IBIF bit gets set in IBSR register. To find its source, check if:
 - TCF = 1 that is reception is complete.
 - IBB= 0 in IBSR register i.e. Bus has transitioned from Busy to Idle state.
 - Ignore Arbitration Loss (IBAL = 1) for IBB = 1.
 - Ignore No Acknowledge condition (RXAK = 1) for receiver.
- i) Read the Data Register (IBDR) to determine data received from Master.

Sequence followed in case of Slave Transmit (Steps a-d of Slave Receive for Address Detect, followed by a-f of Master Transmit for Data Transmit).

Sequence followed in case of Master Receive (Steps a-f of Master Transmit for Address dispatch, followed by e-h of Slave Receive for Data Receive)

52.6.3.3 Generating STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a STOP condition is generated by a master transmitter.

```

if (tx_count == 0) or// check to see if all data bytes have been
transmitted
    (bit 0, IBSR == 1) {// or if no ACK generated
        clear bit 5, IBCR// generate stop condition
    }
else {
    IBDR = data_to_transmit// write byte of data to DATA register
    tx_count --// decrement counter
} // return from interrupt

```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the NOACK bit in IBCR before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```

rx_count --// decrease the rx counter
if (rx_count ==1)// 2nd last byte to be read ?

```

```
    bit 3, IBCR = 1// disable ACK
if (rx_count == 0)// last byte to be read ?
    bit 5, IBCR = 0// generate stop signal
else
    data_received = IBDR// read RX data and store
```

52.6.3.4 Generating repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
    bit 2, IBCR = 1// generate another start (restart)
    IBDR == calling_address// transmit the calling address
```

52.6.3.5 Loss of arbitration

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

52.6.4 Programming guidelines specific to slave mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an “end of data” signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

52.6.5 DMA application information

The DMA interface on the I²C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the DMA enable bit in the control register is not set when the I²C module is configured in slave mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I²C block.

The CPU should also keep the I²C interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The DMAEN bit in the IBCR register works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, dependant on the setting of the DMAEN bit. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the I²C should never be configured for slave operation.

The following sections detail how to set up a DMA transfer and what intervention is required from the CPU. It is assumed that the system DMA controller is capable of generating an interrupt after a certain number of DMA transfers have taken place. The sections present the following topics:

- [Section 52.6.5.1, DMA mode, master transmit](#)
- [Section 52.6.5.2, DMA mode, master reception](#)
- [Section 52.6.5.3, Exiting DMA mode, system requirement considerations](#)

52.6.5.1 DMA mode, master transmit

The following flow diagram details exactly the operation for using a DMA controller to transmit “n” data bytes to a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the last data byte) can be transferred by the DMA controller. The last data byte must be transferred by the CPU.

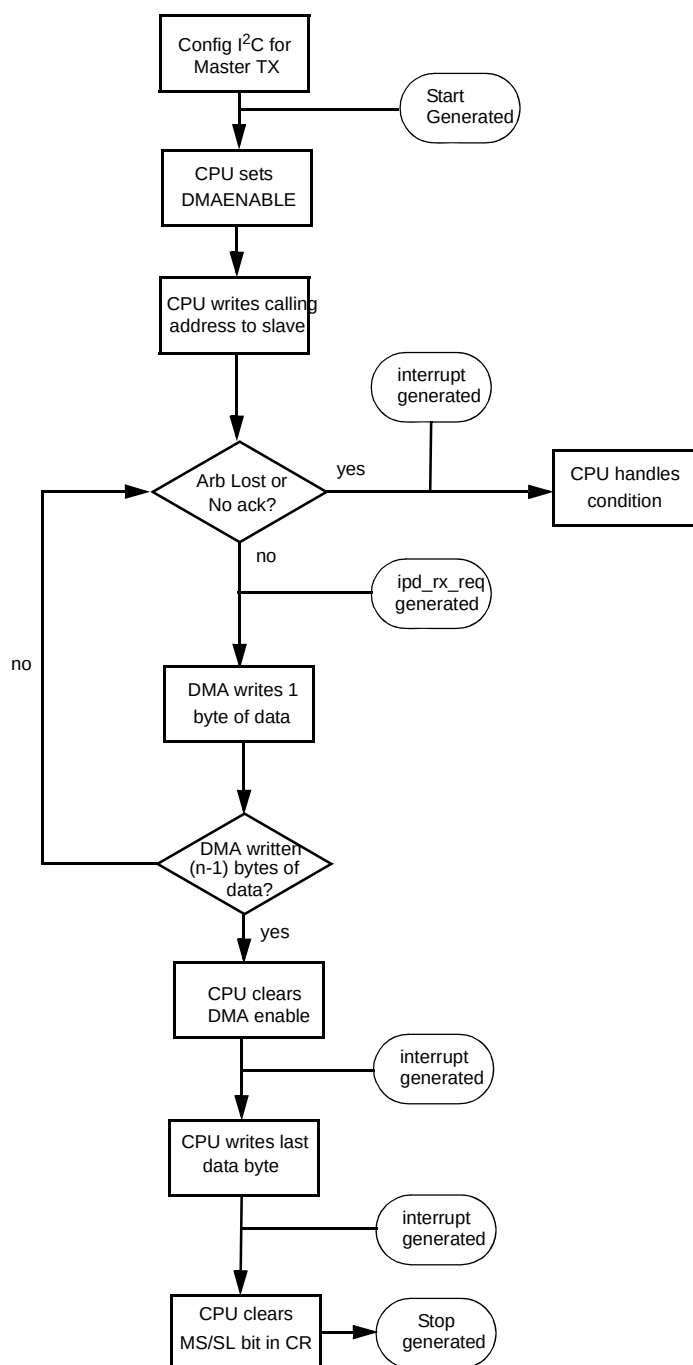


Figure 1122. Flow-Chart of DMA mode master transmit

52.6.5.2 DMA mode, master reception

The following flow diagram details the exact operation for using a DMA controller to receive “n” data bytes from a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the two last data bytes) can be read by the DMA controller. The last two data bytes must be transferred by the CPU.

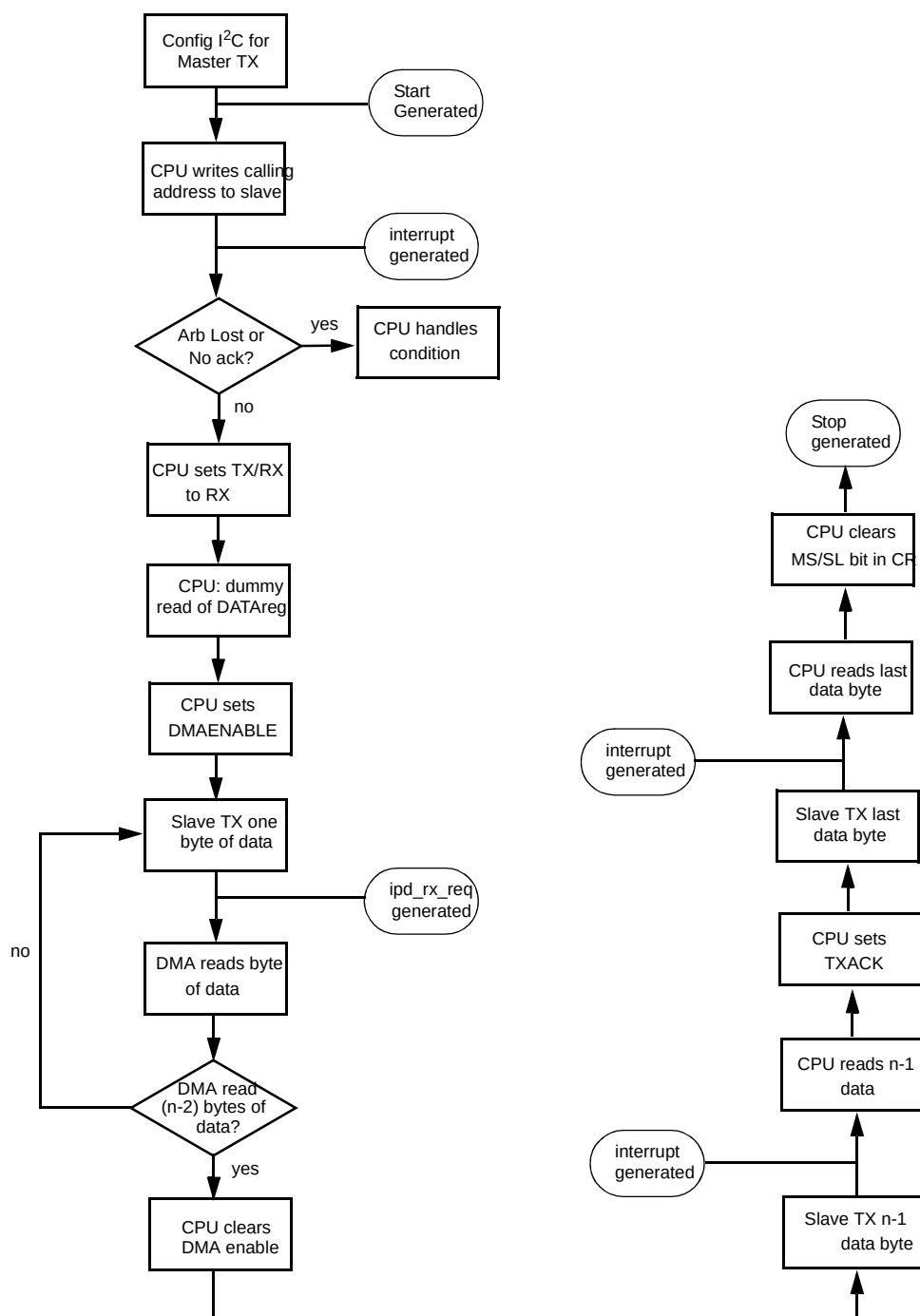


Figure 1123. Flow-Chart of DMA mode master receive

52.6.5.3 Exiting DMA mode, system requirement considerations

As described above, the final transfers of both TX and RX transfers need to be handled via interrupt by the CPU. To change from DMA to interrupt driven transfers in the I²C module, you have to disable the DMAEN bit in the IBCR register. The trigger to exit the DMA mode is

that the programmed DMA Transfer Control Descriptor (TCD) has completed all its transfers to/from the I²C module.

After the last DMA write (TX mode) to the I²C the module will immediately start the next I²C-bus transfer. The same is true for receive mode. After the DMA read from the IBDR register the module initiates the next I²C-bus transfer. This results in two possible scenarios in the DMA mode exiting scheme.

1. Fast reaction

The DMAEN bit is cleared before the next I²C-bus transfer completes. In this case the module will raise an interrupt request to the CPU which can be serviced normally.

2. Slow reaction

The DMAEN bit is cleared after the next I²C-bus transfer has already completed. In this case, the module will not raise an interrupt request to the CPU. Instead the TCF bit can be read to determine that the transfer completed and the module is ready for further transfer.

What is fast/slow reaction?

The reaction time T_R for the system to disable DMAEN after the last DMA controller access to the I²C is the time required for one byte transfer over the I²C. For 'fast reaction' the disabling has to occur before the 9th bit of the data transfer which is the ACK bit. So the time available is eight times the SCL period.

Equation 77 $T_R = 8 \times T_{SCL}$

In fast mode, with 400 kbit/s, T_{SCL} is 2.5 μ s, so T_R is 20 μ s.

Depending on the system and DMA controller there are different possibilities for the de-assertion of DMAEN. Three options are:

1. CPU intervention via Interrupt

The DMA controller is programmed to signal an interrupt to the CPU which is then responsible for the de-assertion of DMAEN. This scheme should be supported by most systems but it can result in a slow reaction time if other higher priority interrupts interfere. Therefore the interrupt handling routine can become complicated as it has to check which of the two cases happened (check TCF bit) and act accordingly. In case of slow reaction you can force an interrupt for the I²C in the interrupt controller to have the further transfer handled by the normal I²C interrupt routine.

Note: The use of nested interrupts can still cause potential issues in this scenario, if someone tries to stall the DMA interrupt between the de-assertion and DMAEN bit and checks the TCF bit.

2. DMA channel linking (if supported)

The Transfer control descriptor in the DMA controller that performs the data transfer is linked to another channel that does a write to the I²C IBCR register to disable the DMAEN bit. This is probably the fastest system solution, but it uses two DMA channels.

Note: Here you have to make sure on system level that no higher priority DMA requests occur between the two linked TCDs as those could again create a scenario of slow reaction.

3. DMA scatter/gather process (if supported)

The Transfer control descriptor in the DMA controller that performs the data transfer has the scatter/gather feature activated. This feature will initiate a reload of another TCD from system RAM after the completion of the first TCD. The new TCD will have its start bit already set and immediately start the required write to the I²C IBCR register to disable the DMAEN bit. This TCD also has scatter/gather activated and is programmed

to reload the initial TCD upon completion, bringing the system back into a “ready-for-I²C-transfer” state. The advantage over the two other solutions is that this neither requires CPU intervention nor a 2nd DMA channel. This comes at the cost of 64 bytes RAM (two TCDs), some system bus transfer overhead and a little increase in application code complexity.

Note: Here you have to make sure at system level that no higher priority DMA requests occur during the scatter/gather process, as those could again create a scenario of slow reaction.

Example latencies for a 32 MHz system with a full speed 32-bit AHB bus and an I²C connected via half speed IPI bus:

- Accessing the I²C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I²C could be faster):

Equation 78 $4 \times T_{IPI} = 4 / 16 \text{ MHz} = 250 \text{ ns}$

- Reloading a new TCD (8 x 32-bit) via AHB to the DMA controller (scatter/gather process):

Equation 79 $8 \times T_{AHB} = 8 / 32 \text{ MHz} = 250 \text{ ns}$

With the DMA scatter/gather process the required IBCR access can be done in 0.5 μs, leaving a large margin of 19.5 μs for additional system delays. In this way, the slow reaction case can be prevented. The system user needs to decide which usage model best suits his overall requirement.

53 Peripheral Sensor Interface (PSI5)

53.1 Introduction

The PSI5 interface offers power supply and data transfer between the application and multiple sensors over a two-wire interface. Data transmission from the application to the sensor or sensors is achieved through modulation of the power supply voltage (sync pulse). Data reception from the sensor or sensors to the application is achieved through modulation of the current drawn by a sensor using Manchester encoding.

At the system level, the PSI5 interface consists of mechanisms integrated on the microcontroller and an external physical layer that provides the high-voltage supply and converts the current drawn by the sensor into a digital Manchester-encoded data stream.

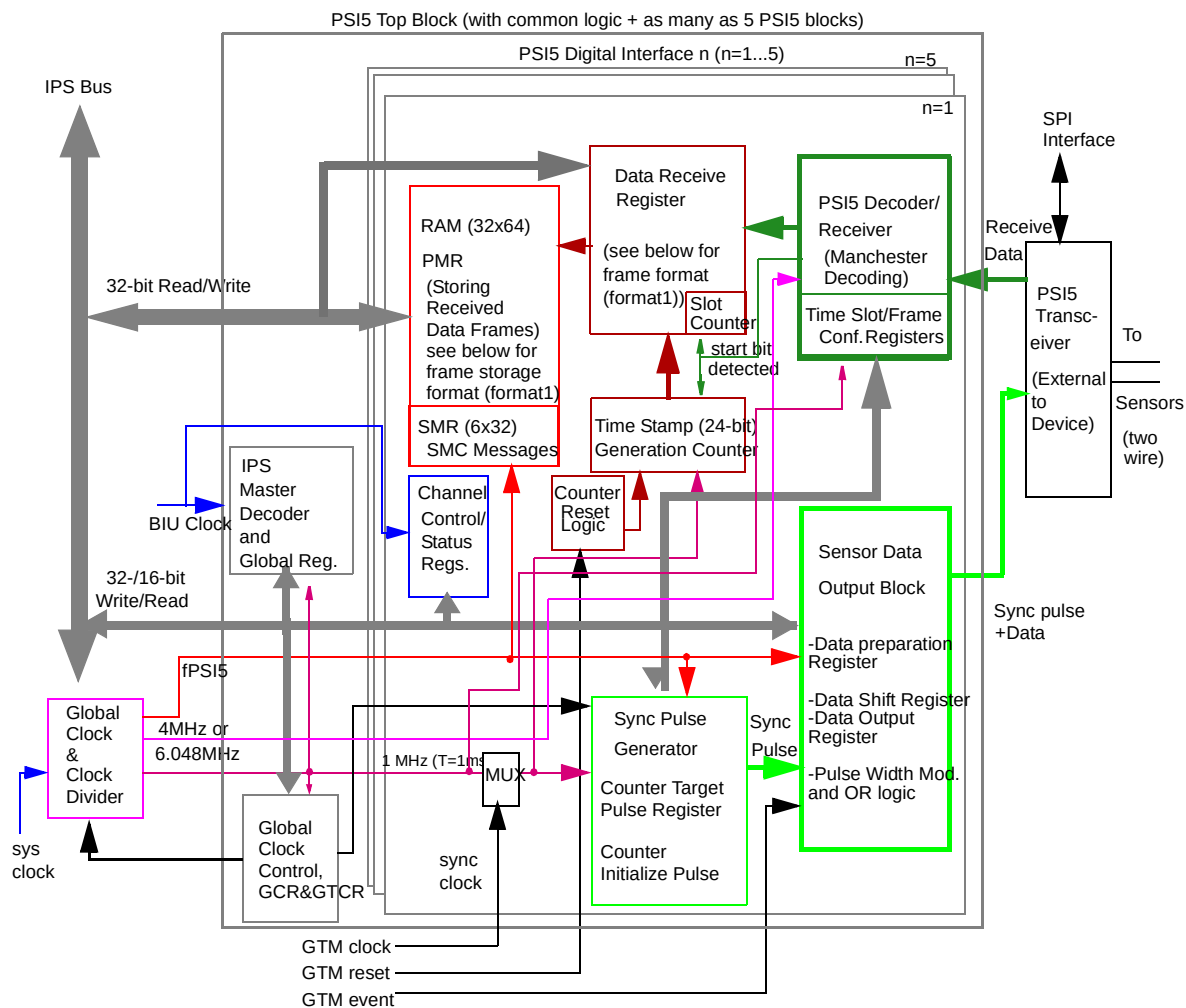
53.1.1 Overview

The PSI5 is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of safety airbag systems. The technical characteristics, the low implementation overhead, as well as the attractive cost make the PSI5 also suitable for many other automotive sensor applications. The development goal of the PSI5 is a flexible, reliable communication standard for automotive sensor applications that can be used and implemented free of charge.

The main features of the PSI5 are high-speed and high-reliability data transfer at the lowest possible implementation overhead cost, offering a universal and flexible solution for multiple sensor applications. The main features of PSI5 standard are:

- Two-wire current interface
- Manchester-coded digital data transmission from sensor to ECU
- High data transmission speed of 125 Kbit/s or optional 189 Kbit/s
- High EMC robustness and low emission
- Wide range of sensor supply current
- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication
- In short, PSI5 standardizes the low-level communication between peripheral sensors and electronic control units. Note that the physical layer for the IP (transceiver) is to be implemented separately outside the IP.

Figure 1124. PSI5 digital interface block diagram



Format1: Stored in RAM Registers

O	F	E	M	E	T	Slot Counter	Time Stamp	Header+ Sensor Data(i)	CRC or Parity bits	C
---	---	---	---	---	---	--------------	------------	------------------------	--------------------	---

1. "C" is set to 1 when CRC or Parity check identifies Error.
2. "T" is set to 1 at timing errors of the incoming frame.
3. "E&EM" is set to 1 at electric errors at any bit PSI5/SMC
4. "F" is set to 1 if no frame received in the slot.
5. "O" is set to 1 if any new frame overwrites prev Unread.Fr.
6. "Slot Counter" Gives the slot number in which the start bits of the frame were detected.
7. "Sensor Data" are stored in RAM automatically after all data bits are received in data receive register.
8. "Time Stamp" is appended to received frame in once start bits are received or rise edge of sync pulse.

53.1.2 Features

- IPS interface for register read/write (big endian)
- Supports bidirectional full duplex operations, but with differing baud rate—low, ECU-to-sensor, fast sensor to ECU
- Manchester decoder operating at both 125 Kbit/s and 189 Kbit/s bit rates with $\times 32$ sample clock rate
- One 64-bit sensor data receive register to which the decoded received sensor data—including header, data, timestamp, frame counter, status bits (E, EM, T, C, F) and CRC, but not including start bits—are written
- A 24-bit time stamp register with a 1 μ s (or the GTM clock) resolution. When first rising edge of start bits (S0) of a message or the rising edge of the SYNC pulse is recognized, the value in the time stamp register will be written to the data receive register
- Provision of resetting the Time Stamp Counter through the reset from the GTM module
- Single 32×64 -bit RAM buffer for storing (moving) content of the data receive register once end of message is recognized
 - Related to each RAM buffer are two diagnostic registers, N and EI. Register N is used to identify the occurrence of new messages in the corresponding RAM buffer, while register EI identifies the error status of these messages. The errors that can set the EI register bits are the F (no frame), E (electrical error), EM (electrical error for SMC bits embedded in PSI5 messages), T (Timing bit error)

and C (CRC error). Which of these errors actually set the EI register bits is configurable.

- DMA support for transferring RAM data (including diagnostic data) to system memory
- DMA transfer through IPS bus
- Support for optional serial messaging channel (SMC) and additional logic to extract SMC frames from PSI5 frames
- Operating modes supported:
 - Asynchronous mode (sensor periodically sends data, no sync pulse, one sensor per interface)
 - Synchronous mode (sync pulse generated, multiple sensor per interface)
- Sync pulse generation for synchronous modes
 - The start of each time slot (1 to 6) (see [Figure 1171](#)) on the bus (after the end of the sync pulse) can be defined with a 1 μ s resolution
 - The end of the last time slot on the bus can be defined with a 1 μ s resolution, up to a maximum sync period time of 16 ms
 - Trigger logic to generate sync pulses
 - 16-bit Channel Trigger Counter (CTC) used to generate the sync pulses by comparing the values with CIPR and CTPR
 - Counter Initialize Pulse Register (CIPR) to allow configuration of initialization (offset) period
 - Channel Trigger Period (CTP) register to allow configuration of sync period. Value in CTP is the value to which the CTC is updated after each sync pulse generation
 - Possibility to have a common 1 MHz clock source
- An upstream register bit chain consisting of Preparation Register, Buffer Register, Shift register (output data) and pulse width modulation logic, is implemented. Using this, it is possible to send a data word to the sensors through use of data sync pulses: logic 1 with programmable data sync pulse width W1, logic 0 with programmable data sync pulse width W0. The data sync pulse timing can be configured to be dependent on the periodic or event driven timing SYNC pulse. Further, there is no difference between timing sync pulse and data sync pulse during transmission (ECU-to-sensor communication).

53.1.3 Modes of operation

53.1.3.1 Disable mode

Disable mode in the PSI5 reduces power consumption. This mode is entered by setting GCR[GLOBAL_DISABLE_REQ]. All the channels enter Disable mode at once when this bit is set, irrespective of the setting of PCCR[PSI5_CH_EN] and PCCR[PSI5_CH_CONFIG]. In this mode the RX, TX and common time base counters are disabled for any operation. This is the default state after module reset is released. PSI5 can exit Disable mode by clearing the GLOBAL_DISABLE_REQ bit and enabling individual PSI5 channels (setting PCCR[PSI5_CH_EN]). Note that when entering Disable mode from any other mode, the register contents are retained and are not changed. The time stamp counter AND the internal sync pulse generator are, however, reset when entering Disable mode.

There is no software reset bit in the PSI5 module. The same has to be managed outside the IP at the system level. Channels will enter “Configuration/Normal mode” based on the value of PCCR[PSI5_CH_CONFIG].

53.1.3.2 Initialization/Configuration mode

This is the configuration phase, in which all PSI5 global, channel specific configuration/control registers can be written. The receiver and transmitter sub-blocks are disabled in this mode. The message receive registers/buffers can be accessed in this mode. This mode can be entered by clearing GCR[GLOBAL_DISABLE_REQ] and setting PCCR[PSI5_CH_CONFIG] and PCCR[PSI5_CH_EN]. All the channel specific control registers can be programmed only in the CONFIG mode. This prevents the software from accidentally causing a change in any critical register.

53.1.3.3 Normal mode (NORMAL mode)

In Normal mode, the receiver and the transmitter blocks become active, error handling is enabled and all the PSI5 protocol functions are also enabled. In this mode, all the PSI5 global and channel-specific configuration registers are locked for writing, although these registers can still be read. Data Preparation Register, Shift Register and data output registers are writable in this mode. Apart from them, there are some register bits that can be written in this mode also. Please refer to the detailed register description. This mode can be entered clearing GCR[GLOBAL_DISABLE_REQ], clearing PCCR[PSI5_CH_CONFIG], and setting PCCR[PSI5_CH_EN].

53.1.3.3.1 PSI5 topology

The different PSI5 user operation modes define topology and parameters of the communication between ECU and sensors such as communication mode, number of data bits, error detection, cycle time, number of time slots per cycle, and bit rate.

These modes are generally denoted as

PSI5-(A / P / U / D)-dd-(P / CRC)-tttt / n(L / H)

[Table 1128](#) provides a key for these notations.

Table 1128. Normal operating states

Communication modes		Description
A	Asynchronous mode	PSI5-A describes a point-to-point connection for unidirectional, asynchronous data transmission. Each sensor is connected to the ECU by two wires. After switching on the power supply, the sensor starts transmitting data to the ECU periodically. Timing and repetition rate of the data transmission are controlled by the sensor. No Sync pulse is sent from ECU side.
P	Synchronous Parallel Bus mode	The synchronous operation modes work according to the TDMA method (Time Division Multiple Access). Each sensor is connected to the ECU by two wires. The sensor data transmission is synchronized by the ECU using voltage modulation. Synchronization (sync pulses) can be optionally used for point-to-point configurations; it is mandatory for bus modes. The sensors can be arranged as parallel bus, universal bus and daisy chain topology based on sensor addressing and interchangeability of sensors.
U	Synchronous Universal Bus mode	
D	Synchronous Daisy Chain Bus mode	

Table 1128. Normal operating states

Communication modes		Description
Data Bits		
dd	No. of data bits	
Error Detection		
P	1 parity bit	
CRC	3-bit Cyclic Redundancy Check (CRC)	
Cycle Time		
tttt	Cycle Time in μs	
n	Number of time slots per cycle	
Bit Rate		
L	125 Kbit/s	
H	189 Kbit/s	

53.1.3.4 Stop mode

When the Stop mode request is asserted, and if there is no active PSI5 transaction ongoing, then the acknowledgement to stop the clocks is given immediately. If start bits of any PSI5 packet have been detected (successful detection of both S0 and S1) then the concerned PSI5 packet is successfully stored and then the Stop mode acknowledgement is generated by the IP. Note that for generating the acknowledgement, the IP does not wait for the complete reception of any ongoing SMC message, but only waits for the complete reception of any ongoing PSI5 packet transaction.

53.1.3.5 Debug mode

In order to enter this mode the debugger has to be connected and it has to encounter a breakpoint. In addition, the `DEBUG_EN` bit has to be written to 1. The `DEBUG_EN` bit can be written by either the CPU or by the debugger but only in Configuration mode. Once the above conditions are satisfied the IP enters the Debug Mode. There are two debugging modes available. For simplicity these are being referred to as the “Freeze” and “Open” modes. The IP enters one of the above debugging modes, based on the configuration of a register bit `DEBUG_FREEZE_CTRL` mentioned in the [Section 53.3.2.2, PSI5 Channel Control Register \(PSI5_PCCR\)](#).

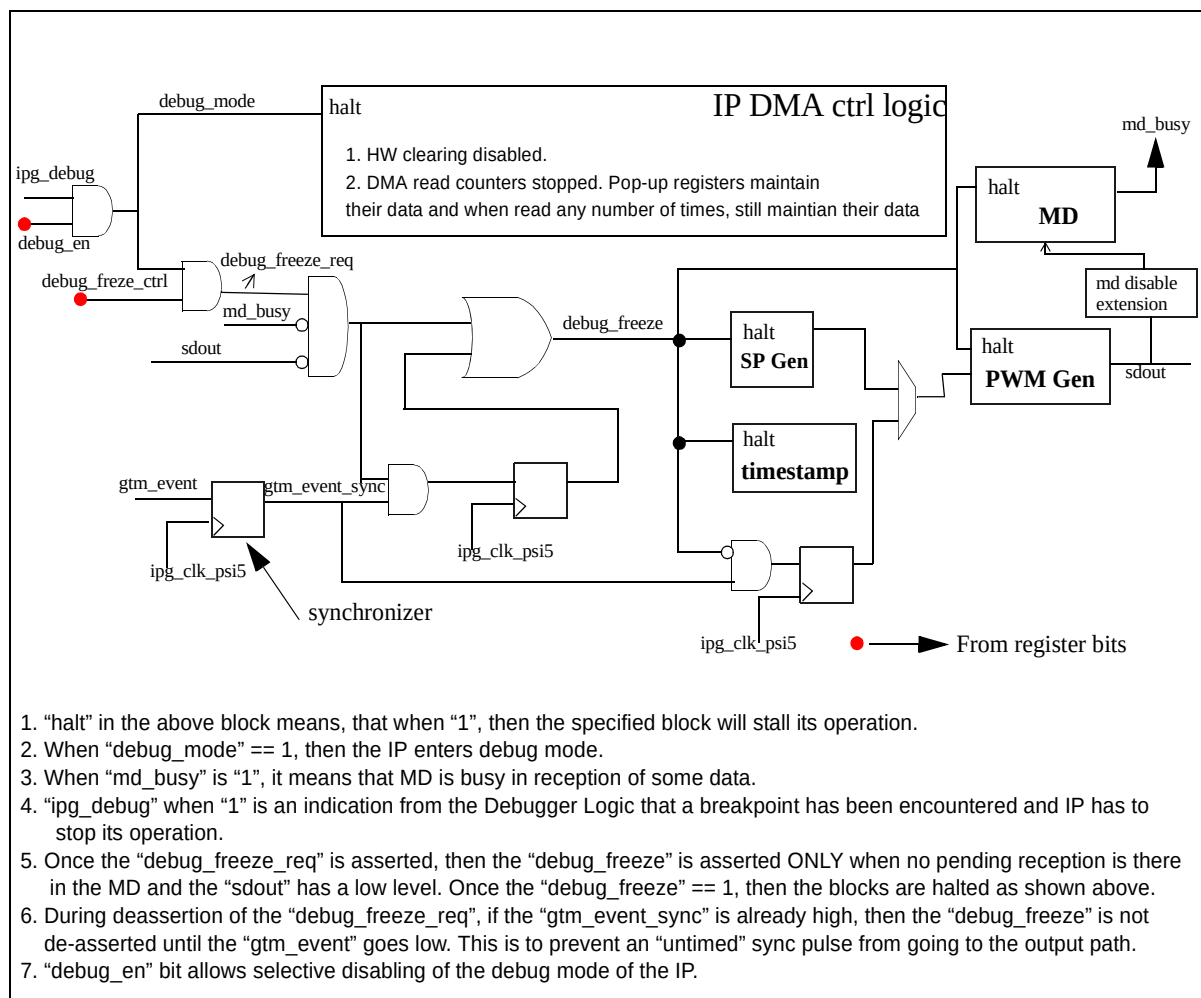
In “Open” mode, the core of the IP keeps on working normally and all the registers are updated from the hardware; just like they were being done in the functional mode (when the IP was out of the debug mode). Thus, even a long term entry to the Debug Mode, won't affect the normal operation of the various registers that are to be updated by the hardware. Down side is that the IP can keep on changing the internal status bits and the registers as the time progresses.

In “Freeze” mode, the core of the IP, the output and the input paths are all stopped coherently. Once the input and the output paths are stopped coherently, then all the internal status bits and registers maintain their status and value irrespective of the time progression. Coherent halting of the input path would mean that there is no active reception ongoing in

the Manchester decoder. Thus, if the Manchester decoder has already detected valid start bits(S0 and S1) and then there is a request to enter the “Freeze” mode, then the current reception is completed before the IP enters the Freeze mode. Coherent halting of the output path would mean that the “sdout” output pin of the IP should be at a logic 0. Once the above conditions are satisfied, only then can the IP enter the “Freeze” mode. Once the IP enters the Freeze mode, the time stamp counter is also halted.

In both the above modes, the various interface registers would be completely visible to the debugger, but debugger action would have no impact on the various status bits and counters that normally get affected by a CPU/DMA read. In this mode, the read pointers of the FIFO do not get incremented, when being read through the Pop-up registers. Thus in Debug mode, reading the DMA PSI5 Message Register (PSI5_DPMR), DMA SMC Frame Register (PSI5_DSFR) or the DMA Diagnostic Status Register (PSI5_DDSR) would keep on returning the same data. Hence the interrupts related to DMA completion operation would not get generated. Further the hardware clearing related to FAST_CLR_SMC and FAST_CLR_PSI5 bits would not have any effect and the hardware clearing would remain disabled. This is also highlighted in [Figure 1125](#).

Figure 1125. Block Diagram of Debug Mode

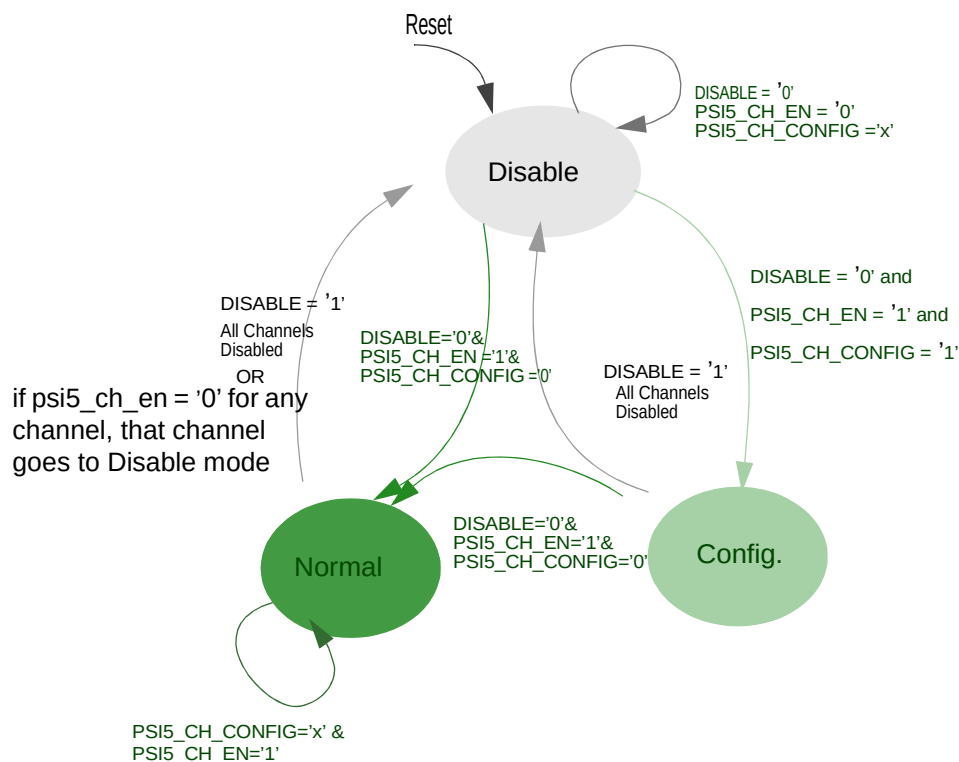


The debugger can write to all the registers just like the CPU can. However, no special privilege is given to the debugger as compared to the CPU. The read/write behavior of the debugger for various registers in Normal/Configuration mode is identical to the behavior of the CPU. Note that the various status bit and counter protections become active ONLY in Debug mode.

53.1.3.6 Operating modes

Figure shows the operating modes and Table 1129 shows the state transition.

Figure 1126. Operating modes



Note Not possible to reconfigure the PSI5 on the fly. A direct transition from the Normal to Config mode is not there to consider safety aspects.

Table 1129. State transition table

GCR [GLOBAL_DISABLE_REQ]	PCCR		State	
	[PSI5_CH_CONFIG]	[PSI5_CH_EN]	Present	Next
1	x	x	x	All channel Disable
0	x	0	x	Specific channel Disable
0	1	1	Disable	Config
0	0	1	Config	Normal

Table 1129. State transition table

GCR [GLOBAL_DISABLE_REQ]	PCCR		State	
	[PSI5_CH_CONFIG]	[PSI5_CH_EN]	Present	Next
0	x	1	Normal	Normal
0	0	1	Disable	Normal

53.2 External signal description

Table 1130. PSI5 port list

Name	Function	I/O
Signals to/from pads		
SDINn_PSI5_m ⁽¹⁾	Digital sensor data input pin of PSI5 module	I
SDOUTn_PSI5_m ⁽¹⁾	Digital sensor data output pin of PSI5 module (sync pulse + data)	O

1. “n” is the Channel Number and “m” is the Module Instance Number.

53.3 Memory map and register description

53.3.1 Memory map

[Table 1131](#) shows the memory map.

Table 1131. PSI5 memory map

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0000	Global Control Register (PSI5_GCR)	R/W	R/W	R/W	— ⁽⁴⁾	Saf- Relv	<i>on page 2136</i>
0x0002— 0x0006	Reserved						
PSI5 Channel 0							
0x0008	PSI5 Channel Control Register (PSI5_CH0_PCCR)	R/W ⁽⁵⁾	R/W	R/W ⁽⁵⁾	0x001F_0000	Saf- Relv	<i>on page 2137</i>
0x000C	DMA Control Register (PSI5_CH0_DCR)	R/W ⁽⁵⁾	R/W	R/W ⁽⁵⁾	0x0000_0000	Saf- Relv	<i>on page 2143</i>

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0010	DMA Status Register (PSI5_CH0_DSR)	R	R/W1c	R/W1c	0x0000_0000	No-Saf	on page 2146
0x0014	General Interrupt Control Register (PSI5_CH0_GICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2147
0x0018	New Data Interrupt Control Register (PSI5_CH0_NDICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2149
0x001C	Overwrite Interrupt Control Register (PSI5_CH0_OWICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2150
0x0020	Error Interrupt Control Register (PSI5_CH0_EICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2150
0x0024	General Interrupt Status Register (PSI5_CH0_GISR)	R	R/W1c/ HC ⁽⁶⁾	R/W1c/ HC ⁽⁶⁾	0x0007_0000	No-Saf	on page 2151
0x0028	DMA PSI5 Message Register (PSI5_CH0_DPMR)	R	R	R	0x0000_0000	Saf-Relv	on page 2154
0x002C	DMA SMC Frame Register (PSI5_CH0_DSFR)	R	R	R	0x0000_0000	Saf-Relv	on page 2155
0x0030	DMA Diagnostic Status Register (PSI5_CH0_DDSR)	R	R	R	0x0000_0000	Saf-Relv	on page 2155
0x0034	PSI5 Message Receive Register Low (PSI5_CH0_PMRRLL)	R	R	R	0x0000_0000	Saf-Relv	on page 2156
0x0038	PSI5 Message Receive Register High (PSI5_CH0_PMRRLH)	R	R	R	0x0000_0000	Saf-Relv	on page 2157
0x003C	PSI5 Message Register 0 Low Word of Data Frame (PSI5_CH0_PMRL0)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0040	PSI5 Message Register 0 High Word of Data Frame (PSI5_CH0_PMRL0)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0044	PSI5 Message Register 1 Low Word of Data Frame (PSI5_CH0_PMRL1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0048	PSI5 Message Register 1 High Word of Data Frame (PSI5_CH0_PMRL1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x004C	PSI5 Message Register 2 Low Word of Data Frame (PSI5_CH0_PMRL2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0050	PSI5 Message Register 2 High Word of Data Frame (PSI5_CH0_PMRL2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0054	PSI5 Message Register 3 Low Word of Data Frame (PSI5_CH0_PMRL3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0058	PSI5 Message Register 3 High Word of Data Frame (PSI5_CH0_PMRL3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x005C	PSI5 Message Register 4 Low Word of Data Frame (PSI5_CH0_PMRL4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0060	PSI5 Message Register 4 High Word of Data Frame (PSI5_CH0_PMRL4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0064	PSI5 Message Register 5 Low Word of Data Frame (PSI5_CH0_PMRL5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0068	PSI5 Message Register 5 High Word of Data Frame (PSI5_CH0_PMRL5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x006C	PSI5 Message Register 6 Low Word of Data Frame (PSI5_CH0_PMRL6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0070	PSI5 Message Register 6 High Word of Data Frame (PSI5_CH0_PMRL6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0074	PSI5 Message Register 7 Low Word of Data Frame (PSI5_CH0_PMRL7)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0078	PSI5 Message Register 7 High Word of Data Frame (PSI5_CH0_PMRL7)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x007C	PSI5 Message Register 8 Low Word of Data Frame (PSI5_CH0_PMRL8)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0080	PSI5 Message Register 8 High Word of Data Frame (PSI5_CH0_PMRL8)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0084	PSI5 Message Register 9 Low Word of Data Frame (PSI5_CH0_PMRL9)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0088	PSI5 Message Register 9 High Word of Data Frame (PSI5_CH0_PMRL9)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x008C	PSI5 Message Register 10 Low Word of Data Frame (PSI5_CH0_PMRL10)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0090	PSI5 Message Register 10 High Word of Data Frame (PSI5_CH0_PMRL10)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0094	PSI5 Message Register 11 Low Word of Data Frame (PSI5_CH0_PMRL11)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0098	PSI5 Message Register 11 High Word of Data Frame (PSI5_CH0_PMRL11)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x009C	PSI5 Message Register 12 Low Word of Data Frame (PSI5_CH0_PMRL12)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x00A0	PSI5 Message Register 12 High Word of Data Frame (PSI5_CH0_PMRL12)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x00A4	PSI5 Message Register 13 Low Word of Data Frame (PSI5_CH0_PMRL13)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x00A8	PSI5 Message Register 13 High Word of Data Frame (PSI5_CH0_PMRL13)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x00AC	PSI5 Message Register 14 Low Word of Data Frame (PSI5_CH0_PMRL14)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x00B0	PSI5 Message Register 14 High Word of Data Frame (PSI5_CH0_PMRL14)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x00B4	PSI5 Message Register 15 Low Word of Data Frame (PSI5_CH0_PMRL15)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x00B8	PSI5 Message Register 15 High Word of Data Frame (PSI5_CH0_PMRL15)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x00BC	PSI5 Message Register 16 Low Word of Data Frame (PSI5_CH0_PMRL16)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x00C0	PSI5 Message Register 16 High Word of Data Frame (PSI5_CH0_PMRL16)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x00C4	PSI5 Message Register 17 Low Word of Data Frame (PSI5_CH0_PMRL17)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x00C8	PSI5 Message Register 17 High Word of Data Frame (PSI5_CH0_PMRL17)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x00CC	PSI5 Message Register 18 Low Word of Data Frame (PSI5_CH0_PMRL18)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x00D0	PSI5 Message Register 18 High Word of Data Frame (PSI5_CH0_PMRL18)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x00D4	PSI5 Message Register 19 Low Word of Data Frame (PSI5_CH0_PMRL19)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x00D8	PSI5 Message Register 19 High Word of Data Frame (PSI5_CH0_PMRL19)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x00DC	PSI5 Message Register 20 Low Word of Data Frame (PSI5_CH0_PMRL20)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x00E0	PSI5 Message Register 20 High Word of Data Frame (PSI5_CH0_PMRL20)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x00E4	PSI5 Message Register 21 Low Word of Data Frame (PSI5_CH0_PMRL21)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x00E8	PSI5 Message Register 21 High Word of Data Frame (PSI5_CH0_PMRL21)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x00EC	PSI5 Message Register 22 Low Word of Data Frame (PSI5_CH0_PMRL22)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x00F0	PSI5 Message Register 22 High Word of Data Frame (PSI5_CH0_PMRL22)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x00F4	PSI5 Message Register 23 Low Word of Data Frame (PSI5_CH0_PMRL23)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x00F8	PSI5 Message Register 23 High Word of Data Frame (PSI5_CH0_PMRL23)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x00FC	PSI5 Message Register 24 Low Word of Data Frame (PSI5_CH0_PMRL24)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0100	PSI5 Message Register 24 High Word of Data Frame (PSI5_CH0_PMRL24)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0104	PSI5 Message Register 25 Low Word of Data Frame (PSI5_CH0_PMRL25)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0108	PSI5 Message Register 25 High Word of Data Frame (PSI5_CH0_PMRL25)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x010C	PSI5 Message Register 26 Low Word of Data Frame (PSI5_CH0_PMRL26)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0110	PSI5 Message Register 26 High Word of Data Frame (PSI5_CH0_PMRL26)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0114	PSI5 Message Register 27 Low Word of Data Frame (PSI5_CH0_PMRL27)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0118	PSI5 Message Register 27 High Word of Data Frame (PSI5_CH0_PMRL27)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x011C	PSI5 Message Register 28 Low Word of Data Frame (PSI5_CH0_PMRL28)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0120	PSI5 Message Register 28 High Word of Data Frame (PSI5_CH0_PMRL28)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0124	PSI5 Message Register 29 Low Word of Data Frame (PSI5_CH0_PMRL29)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0128	PSI5 Message Register 29 High Word of Data Frame (PSI5_CH0_PMRL29)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x012C	PSI5 Message Register 30 Low Word of Data Frame (PSI5_CH0_PMRL30)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0130	PSI5 Message Register 30 High Word of Data Frame (PSI5_CH0_PMRL30)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0134	PSI5 Message Register 31 Low Word of Data Frame (PSI5_CH0_PMRL31)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0138	PSI5 Message Register 31 High Word of Data Frame (PSI5_CH0_PMRL31)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x013C	SMC Frame RX Register 1 (PSI5_CH0_SFR1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x0140	SMC Frame RX Register 2 (PSI5_CH0_SFR2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x0144	SMC Frame RX Register 3 (PSI5_CH0_SFR3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x0148	SMC Frame RX Register 4 (PSI5_CH0_SFR4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x014C	SMC Frame RX Register 5 (PSI5_CH0_SFR5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x0150	SMC Frame RX Register 6 (PSI5_CH0_SFR6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x0154	New Data Status Register (PSI5_CH0_NDSR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2163
0x0158	Over Write Status Register (PSI5_CH0_OWSR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2163
0x015C	Error Status Register (PSI5_CH0_EISR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2164
0x0160	Set New Data Status Register (PSI5_CH0_SNDSR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2165
0x0164	Set Overwrite Status Register (PSI5_CH0_SOWSR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2165
0x0168	Set Error Status Register (PSI5_CH0_SEISR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2166
0x016C	Set SMC Error Status Register (PSI5_CH0_SSESR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2167
0x0170	Sync Pulse Start Time Stamp Read Register (PSI5_CH0_STSRR)	R	R	R	0x0000_0000	Saf-Relv	on page 2168
0x0174	Data Reception Time Stamp Read Register (PSI5_CH0_DTSRR)	R	R	R	0x0000_0000	Saf-Relv	on page 2168
0x0178	Slot 1 Frame Configuration Register (PSI5_CH0_S1FCR)	R	R/W	R	0x0000_0010	Saf-Relv	on page 2169

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x017C	Slot 2 Frame Configuration Register (PSI5_CH0_S2FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x0180	Slot 3 Frame Configuration Register (PSI5_CH0_S3FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x0184	Slot 4 Frame Configuration Register (PSI5_CH0_S4FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x0188	Slot 5 Frame Configuration Register (PSI5_CH0_S5FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x018C	Slot 6 Frame Configuration Register (PSI5_CH0_S6FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x0190	Slot 1 Start Boundary Register (PSI5_CH0_S1SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x0192	Slot 2 Start Boundary Register (PSI5_CH0_S2SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x0194	Slot 3 Start Boundary Register (PSI5_CH0_S3SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x0196	Slot 4 Start Boundary Register (PSI5_CH0_S4SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x0198	Slot 5 Start Boundary Register (PSI5_CH0_S5SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x019A	Slot 6 Start Boundary Register (PSI5_CH0_S6SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x019C	Slot <i>n</i> End Boundary Register (PSI5_CH0_S <i>n</i> EBR)	R	R/W	R	0x0000_0000	Saf- Relv	on page 2171
0x01A0	Data Output Block Configuration Register (PSI5_CH0_DOBCE)	R	R/W/ W1S ⁽⁹⁾	R/W1S ⁽⁹⁾	0x0000	Saf- Relv	on page 2172
0x01A2	Manchester Decoder Disable Offset (PSI5_CH0_MDDIS_OFF)	R	R/W	R	0x0000	Saf- Relv	on page 2175
0x01A4	Pulse Width for Data Bit Value 0 (PSI5_CH0_PW0D)	R	R/W	R	0x0000	Saf- Relv	on page 2176
0x01A6	Pulse Width for Data Bit Value 1 (PSI5_CH0_PW1D)	R	R/W	R	0x0000	Saf- Relv	on page 2177
0x01A8	Counter target Pulse Register (PSI5_CH0_CTPR)	R	R/W	R	0x0000	Saf- Relv	on page 2177
0x01AA	Counter Initialize Pulse Register (PSI5_CH0_CIPR)	R	R/W	R	0x0000	Saf- Relv	on page 2178

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x01AC	Data Preparation Register Low Word (PSI5_CH0_DPRL)	R	R/W ⁽¹⁰⁾	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2179
0x01B0	Data Preparation Register High Word (PSI5_CH0_DPRH)	R	R	R	0x0000_0000	Saf-Relv	on page 2180
0x01B4	Data Buffer Register Low Word (PSI5_CH0_DBRL)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2181
0x01B8	Data Buffer Register (High Word (PSI5_CH0_DBRH)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2185
0x01BC	Data Shift Register Low Word (PSI5_CH0_DSRL)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2187
0x01C0	Data Shift Register High Word (PSI5_CH0_DSRH)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2188
PSI5 Channel 1							
0x01C8	PSI5 Channel Control Register (PSI5_CH1_PCCR)	R/W ⁽⁵⁾	R/W	R/W ⁽⁵⁾	0x001F_0000	Saf-Relv	on page 2137
0x01CC	DMA Control Register (PSI5_CH1_DCR)	R/W ⁽⁵⁾	R/W	R/W ⁽⁵⁾	0x0000_0000	Saf-Relv	on page 2143
0x01D0	DMA Status Register (PSI5_CH1_DSR)	R	R/W1c	R/W1c	0x0000_0000	No-Saf	on page 2146
0x01D4	General Interrupt Control Register (PSI5_CH1_GICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2147
0x01D8	New Data Interrupt Control Register (PSI5_CH1_NDICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2149
0x01DC	Overwrite Interrupt Control Register (PSI5_CH1_OWICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2150
0x01E0	Error Interrupt Control Register (PSI5_CH1_EICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2150
0x01E4	General Interrupt Status Register (PSI5_CH1_GISR)	R	R/W1c/ HC ⁽⁶⁾	R/W1c/ HC ⁽⁶⁾	0x0007_0000	No-Saf	on page 2151
0x01E8	DMA PSI5 Message Register (PSI5_CH1_DPMR)	R	R	R	0x0000_0000	Saf-Relv	on page 2154
0x01EC	DMA SMC Frame Register (PSI5_CH1_DSFR)	R	R	R	0x0000_0000	Saf-Relv	on page 2155
0x01F0	DMA Diagnostic Status Register (PSI5_CH1_DDSDR)	R	R	R	0x0000_0000	Saf-Relv	on page 2155

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x01F4	PSI5 Message Receive Register Low (PSI5_CH1_PMRRL)	R	R	R	0x0000_0000	Saf-Relv	on page 2156
0x01F8	PSI5 Message Receive Register High (PSI5_CH1_PMRRH)	R	R	R	0x0000_0000	Saf-Relv	on page 2157
0x01FC	PSI5 Message Register 0 Low Word of Data Frame (PSI5_CH1_PMRL0)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0200	PSI5 Message Register 0 High Word of Data Frame (PSI5_CH1_PMRL0)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0204	PSI5 Message Register 1 Low Word of Data Frame (PSI5_CH1_PMRL1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0208	PSI5 Message Register 1 High Word of Data Frame (PSI5_CH1_PMRL1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x020C	PSI5 Message Register 2 Low Word of Data Frame (PSI5_CH1_PMRL2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0210	PSI5 Message Register 2 High Word of Data Frame (PSI5_CH1_PMRL2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0214	PSI5 Message Register 3 Low Word of Data Frame (PSI5_CH1_PMRL3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0218	PSI5 Message Register 3 High Word of Data Frame (PSI5_CH1_PMRL3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x021C	PSI5 Message Register 4 Low Word of Data Frame (PSI5_CH1_PMRL4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0220	PSI5 Message Register 4 High Word of Data Frame (PSI5_CH1_PMRL4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0224	PSI5 Message Register 5 Low Word of Data Frame (PSI5_CH1_PMRL5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0228	PSI5 Message Register 5 High Word of Data Frame (PSI5_CH1_PMRL5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x022C	PSI5 Message Register 6 Low Word of Data Frame (PSI5_CH1_PMRL6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0230	PSI5 Message Register 6 High Word of Data Frame (PSI5_CH1_PMRL6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0234	PSI5 Message Register 7 Low Word of Data Frame (PSI5_CH1_PMRL7)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0238	PSI5 Message Register 7 High Word of Data Frame (PSI5_CH1_PMRL7)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x023C	PSI5 Message Register 8 Low Word of Data Frame (PSI5_CH1_PMRL8)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0240	PSI5 Message Register 8 High Word of Data Frame (PSI5_CH1_PMRL8)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0244	PSI5 Message Register 9 Low Word of Data Frame (PSI5_CH1_PMRL9)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0248	PSI5 Message Register 9 High Word of Data Frame (PSI5_CH1_PMRL9)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x024C	PSI5 Message Register 10 Low Word of Data Frame (PSI5_CH1_PMRL10)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0250	PSI5 Message Register 10 High Word of Data Frame (PSI5_CH1_PMRL10)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0254	PSI5 Message Register 11 Low Word of Data Frame (PSI5_CH1_PMRL11)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0258	PSI5 Message Register 11 High Word of Data Frame (PSI5_CH1_PMRL11)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x025C	PSI5 Message Register 12 Low Word of Data Frame (PSI5_CH1_PMRL12)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0260	PSI5 Message Register 12 High Word of Data Frame (PSI5_CH1_PMRL12)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0264	PSI5 Message Register 13 Low Word of Data Frame (PSI5_CH1_PMRL13)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0268	PSI5 Message Register 13 High Word of Data Frame (PSI5_CH1_PMRL13)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x026C	PSI5 Message Register 14 Low Word of Data Frame (PSI5_CH1_PMRL14)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0270	PSI5 Message Register 14 High Word of Data Frame (PSI5_CH1_PMRL14)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0274	PSI5 Message Register 15 Low Word of Data Frame (PSI5_CH1_PMRL15)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0278	PSI5 Message Register 15 High Word of Data Frame (PSI5_CH1_PMRL15)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x027C	PSI5 Message Register 16 Low Word of Data Frame (PSI5_CH1_PMRL16)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0280	PSI5 Message Register 16 High Word of Data Frame (PSI5_CH1_PMRL16)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0284	PSI5 Message Register 17 Low Word of Data Frame (PSI5_CH1_PMRL17)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0288	PSI5 Message Register 17 High Word of Data Frame (PSI5_CH1_PMRL17)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x028C	PSI5 Message Register 18 Low Word of Data Frame (PSI5_CH1_PMRL18)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0290	PSI5 Message Register 18 High Word of Data Frame (PSI5_CH1_PMRL18)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0294	PSI5 Message Register 19 Low Word of Data Frame (PSI5_CH1_PMRL19)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0298	PSI5 Message Register 19 High Word of Data Frame (PSI5_CH1_PMRL19)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x029C	PSI5 Message Register 20 Low Word of Data Frame (PSI5_CH1_PMRL20)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x02A0	PSI5 Message Register 20 High Word of Data Frame (PSI5_CH1_PMRL20)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x02A4	PSI5 Message Register 21 Low Word of Data Frame (PSI5_CH1_PMRL21)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x02A8	PSI5 Message Register 21 High Word of Data Frame (PSI5_CH1_PMRL21)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x02AC	PSI5 Message Register 22 Low Word of Data Frame (PSI5_CH1_PMRL22)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x02B0	PSI5 Message Register 22 High Word of Data Frame (PSI5_CH1_PMRL22)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x02B4	PSI5 Message Register 23 Low Word of Data Frame (PSI5_CH1_PMRL23)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x02B8	PSI5 Message Register 23 High Word of Data Frame (PSI5_CH1_PMRL23)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x02BC	PSI5 Message Register 24 Low Word of Data Frame (PSI5_CH1_PMRL24)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x02C0	PSI5 Message Register 24 High Word of Data Frame (PSI5_CH1_PMRL24)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x02C4	PSI5 Message Register 25 Low Word of Data Frame (PSI5_CH1_PMRL25)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x02C8	PSI5 Message Register 25 High Word of Data Frame (PSI5_CH1_PMRL25)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x02CC	PSI5 Message Register 26 Low Word of Data Frame (PSI5_CH1_PMRL26)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x02D0	PSI5 Message Register 26 High Word of Data Frame (PSI5_CH1_PMRL26)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x02D4	PSI5 Message Register 27 Low Word of Data Frame (PSI5_CH1_PMRL27)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x02D8	PSI5 Message Register 27 High Word of Data Frame (PSI5_CH1_PMRL27)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x02DC	PSI5 Message Register 28 Low Word of Data Frame (PSI5_CH1_PMRL28)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x02E0	PSI5 Message Register 28 High Word of Data Frame (PSI5_CH1_PMRL28)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x02E4	PSI5 Message Register 29 Low Word of Data Frame (PSI5_CH1_PMRL29)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x02E8	PSI5 Message Register 29 High Word of Data Frame (PSI5_CH1_PMRL29)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x02EC	PSI5 Message Register 30 Low Word of Data Frame (PSI5_CH1_PMRL30)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x02F0	PSI5 Message Register 30 High Word of Data Frame (PSI5_CH1_PMRL30)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x02F4	PSI5 Message Register 31 Low Word of Data Frame (PSI5_CH1_PMRL31)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x02F8	PSI5 Message Register 31 High Word of Data Frame (PSI5_CH1_PMRL31)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x02FC	SMC Frame RX Register 1 (PSI5_CH1_SFR1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2161
0x0300	SMC Frame RX Register 2 (PSI5_CH1_SFR2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2161
0x0304	SMC Frame RX Register 3 (PSI5_CH1_SFR3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2161
0x0308	SMC Frame RX Register 4 (PSI5_CH1_SFR4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2161
0x030C	SMC Frame RX Register 5 (PSI5_CH1_SFR5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2161
0x0310	SMC Frame RX Register 6 (PSI5_CH1_SFR6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2161
0x0314	New Data Status Register (PSI5_CH1_NDSR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2163

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0318	Over Write Status Register (PSI5_CH1_OWSR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2163
0x031C	Error Status Register (PSI5_CH1_EISR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2164
0x0320	Set New Data Status Register (PSI5_CH1_SNDSR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2165
0x0324	Set Overwrite Status Register (PSI5_CH1_SOWSR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2165
0x0328	Set Error Status Register (PSI5_CH1_SEISR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2166
0x032C	Set SMC Error Status Register (PSI5_CH1_SSESR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2167
0x0330	Sync Pulse Start Time Stamp Read Register (PSI5_CH1_STSRR)	R	R	R	0x0000_0000	Saf-Relv	on page 2168
0x0334	Data Reception Time Stamp Read Register (PSI5_CH1_DTSRR)	R	R	R	0x0000_0000	Saf-Relv	on page 2168
0x0338	Slot 1 Frame Configuration Register (PSI5_CH1_S1FCR)	R	R/W	R	0x0000_0010	Saf-Relv	on page 2169
0x033C	Slot 2 Frame Configuration Register (PSI5_CH1_S2FCR)	R	R/W	R	0x0000_0010	Saf-Relv	on page 2169
0x0340	Slot 3 Frame Configuration Register (PSI5_CH1_S3FCR)	R	R/W	R	0x0000_0010	Saf-Relv	on page 2169
0x0344	Slot 4 Frame Configuration Register (PSI5_CH1_S4FCR)	R	R/W	R	0x0000_0010	Saf-Relv	on page 2169
0x0348	Slot 5 Frame Configuration Register (PSI5_CH1_S5FCR)	R	R/W	R	0x0000_0010	Saf-Relv	on page 2169
0x034C	Slot 6 Frame Configuration Register (PSI5_CH1_S6FCR)	R	R/W	R	0x0000_0010	Saf-Relv	on page 2169
0x0350	Slot 1 Start Boundary Register (PSI5_CH1_S1SBR)	R	R/W	R	0x0000	Saf-Relv	on page 2170
0x0352	Slot 2 Start Boundary Register (PSI5_CH1_S2SBR)	R	R/W	R	0x0000	Saf-Relv	on page 2170
0x0354	Slot 3 Start Boundary Register (PSI5_CH1_S3SBR)	R	R/W	R	0x0000	Saf-Relv	on page 2170
0x0356	Slot 4 Start Boundary Register (PSI5_CH1_S4SBR)	R	R/W	R	0x0000	Saf-Relv	on page 2170

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0358	Slot 5 Start Boundary Register (PSI5_CH1_S5SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x035A	Slot 6 Start Boundary Register (PSI5_CH1_S6SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x035C	Slot <i>n</i> End Boundary Register (PSI5_CH1_S <i>n</i> EBR)	R	R/W	R	0x0000_0000	Saf- Relv	on page 2171
0x0360	Data Output Block Configuration Register (PSI5_CH1_DOBCR)	R	R/W/ W1S ⁽⁹⁾	R/W1S ⁽⁹⁾	0x0000	Saf- Relv	on page 2172
0x0362	Manchester Decoder Disable Offset (PSI5_CH1_MDDIS_OFF)	R	R/W	R	0x0000	Saf- Relv	on page 2175
0x0364	Pulse Width for Data Bit Value 0 (PSI5_CH1_PW0D)	R	R/W	R	0x0000	Saf- Relv	on page 2176
0x0366	Pulse Width for Data Bit Value 1 (PSI5_CH1_PW1D)	R	R/W	R	0x0000	Saf- Relv	on page 2177
0x0368	Counter target Pulse Register (PSI5_CH1_CTPR)	R	R/W	R	0x0000	Saf- Relv	on page 2177
0x036A	Counter Initialize Pulse Register (PSI5_CH1_CIPR)	R	R/W	R	0x0000	Saf- Relv	on page 2178
0x036C	Data Preparation Register Low Word (PSI5_CH1_DPRL)	R	R/W ⁽¹⁰⁾	R/W ⁽¹⁰⁾	0x0000_0000	Saf- Relv	on page 2179
0x0370	Data Preparation Register High Word (PSI5_CH1_DPRH)	R	R	R	0x0000_0000	Saf- Relv	on page 2180
0x0374	Data Buffer Register Low Word (PSI5_CH1_DBRL)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf- Relv	on page 2181
0x0378	Data Buffer Register (High Word (PSI5_CH1_DBRH)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf- Relv	on page 2185
0x037C	Data Shift Register Low Word (PSI5_CH1_DSRL)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf- Relv	on page 2187
0x0380	Data Shift Register High Word (PSI5_CH1_DSRH)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf- Relv	on page 2188
PSI5 Channel 2							
0x0388	PSI5 Channel Control Register (PSI5_CH2_PCCR)	R/W ⁽⁵⁾	R/W	R/W ⁽⁵⁾	0x001F_0000	Saf- Relv	on page 2137
0x038C	DMA Control Register (PSI5_CH2_DCR)	R/W ⁽⁵⁾	R/W	R/W ⁽⁵⁾	0x0000_0000	Saf- Relv	on page 2143

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0390	DMA Status Register (PSI5_CH2_DSR)	R	R/W1c	R/W1c	0x0000_0000	No-Saf	on page 2146
0x0394	General Interrupt Control Register (PSI5_CH2_GICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2147
0x0398	New Data Interrupt Control Register (PSI5_CH2_NDICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2149
0x039C	Overwrite Interrupt Control Register (PSI5_CH2_OWICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2150
0x03A0	Error Interrupt Control Register (PSI5_CH2_EICR)	R	R/W	R/W	0x0000_0000	Saf-Relv	on page 2150
0x03A4	General Interrupt Status Register (PSI5_CH2_GISR)	R	R/W1c/ HC ⁽⁶⁾	R/W1c/ HC ⁽⁶⁾	0x0007_0000	No-Saf	on page 2151
0x03A8	DMA PSI5 Message Register (PSI5_CH2_DPMR)	R	R	R	0x0000_0000	Saf-Relv	on page 2154
0x03AC	DMA SMC Frame Register (PSI5_CH2_DSFR)	R	R	R	0x0000_0000	Saf-Relv	on page 2155
0x03B0	DMA Diagnostic Status Register (PSI5_CH2_DDSR)	R	R	R	0x0000_0000	Saf-Relv	on page 2155
0x03B0	PSI5 Message Receive Register Low (PSI5_CH2_PMRRL)	R	R	R	0x0000_0000	Saf-Relv	on page 2156
0x03B4	PSI5 Message Receive Register High (PSI5_CH2_PMRRH)	R	R	R	0x0000_0000	Saf-Relv	on page 2157
0x03BC	PSI5 Message Register 0 Low Word of Data Frame (PSI5_CH2_PMRL0)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x03C0	PSI5 Message Register 0 High Word of Data Frame (PSI5_CH2_PMRL0)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x03C4	PSI5 Message Register 1 Low Word of Data Frame (PSI5_CH2_PMRL1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x03C8	PSI5 Message Register 1 High Word of Data Frame (PSI5_CH2_PMRL1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x03CC	PSI5 Message Register 2 Low Word of Data Frame (PSI5_CH2_PMRL2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x03D0	PSI5 Message Register 2 High Word of Data Frame (PSI5_CH2_PMRL2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x03D4	PSI5 Message Register 3 Low Word of Data Frame (PSI5_CH2_PMRL3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x03D8	PSI5 Message Register 3 High Word of Data Frame (PSI5_CH2_PMRL3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x03DC	PSI5 Message Register 4 Low Word of Data Frame (PSI5_CH2_PMRL4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x03E0	PSI5 Message Register 4 High Word of Data Frame (PSI5_CH2_PMRL4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x03E4	PSI5 Message Register 5 Low Word of Data Frame (PSI5_CH2_PMRL5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x03E8	PSI5 Message Register 5 High Word of Data Frame (PSI5_CH2_PMRL5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x03EC	PSI5 Message Register 6 Low Word of Data Frame (PSI5_CH2_PMRL6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x03F0	PSI5 Message Register 6 High Word of Data Frame (PSI5_CH2_PMRL6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x03F4	PSI5 Message Register 7 Low Word of Data Frame (PSI5_CH2_PMRL7)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x03F8	PSI5 Message Register 7 High Word of Data Frame (PSI5_CH2_PMRL7)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x03FC	PSI5 Message Register 8 Low Word of Data Frame (PSI5_CH2_PMRL8)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0400	PSI5 Message Register 8 High Word of Data Frame (PSI5_CH2_PMRL8)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0404	PSI5 Message Register 9 Low Word of Data Frame (PSI5_CH2_PMRL9)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0408	PSI5 Message Register 9 High Word of Data Frame (PSI5_CH2_PMRL9)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x040C	PSI5 Message Register 10 Low Word of Data Frame (PSI5_CH2_PMRL10)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0410	PSI5 Message Register 10 High Word of Data Frame (PSI5_CH2_PMRL10)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0414	PSI5 Message Register 11 Low Word of Data Frame (PSI5_CH2_PMRL11)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0418	PSI5 Message Register 11 High Word of Data Frame (PSI5_CH2_PMRL11)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x041C	PSI5 Message Register 12 Low Word of Data Frame (PSI5_CH2_PMRL12)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0420	PSI5 Message Register 12 High Word of Data Frame (PSI5_CH2_PMRL12)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0424	PSI5 Message Register 13 Low Word of Data Frame (PSI5_CH2_PMRL13)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0428	PSI5 Message Register 13 High Word of Data Frame (PSI5_CH2_PMRL13)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x042C	PSI5 Message Register 14 Low Word of Data Frame (PSI5_CH2_PMRL14)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0430	PSI5 Message Register 14 High Word of Data Frame (PSI5_CH2_PMRL14)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0434	PSI5 Message Register 15 Low Word of Data Frame (PSI5_CH2_PMRL15)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0438	PSI5 Message Register 15 High Word of Data Frame (PSI5_CH2_PMRL15)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x043C	PSI5 Message Register 16 Low Word of Data Frame (PSI5_CH2_PMRL16)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0440	PSI5 Message Register 16 High Word of Data Frame (PSI5_CH2_PMRL16)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0444	PSI5 Message Register 17 Low Word of Data Frame (PSI5_CH2_PMRL17)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0448	PSI5 Message Register 17 High Word of Data Frame (PSI5_CH2_PMRL17)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x044C	PSI5 Message Register 18 Low Word of Data Frame (PSI5_CH2_PMRL18)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0450	PSI5 Message Register 18 High Word of Data Frame (PSI5_CH2_PMRL18)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0454	PSI5 Message Register 19 Low Word of Data Frame (PSI5_CH2_PMRL19)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0458	PSI5 Message Register 19 High Word of Data Frame (PSI5_CH2_PMRL19)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x045C	PSI5 Message Register 20 Low Word of Data Frame (PSI5_CH2_PMRL20)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0460	PSI5 Message Register 20 High Word of Data Frame (PSI5_CH2_PMRL20)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0464	PSI5 Message Register 21 Low Word of Data Frame (PSI5_CH2_PMRL21)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0468	PSI5 Message Register 21 High Word of Data Frame (PSI5_CH2_PMRL21)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x046C	PSI5 Message Register 22 Low Word of Data Frame (PSI5_CH2_PMRL22)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0470	PSI5 Message Register 22 High Word of Data Frame (PSI5_CH2_PMRL22)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x0474	PSI5 Message Register 23 Low Word of Data Frame (PSI5_CH2_PMRL23)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0478	PSI5 Message Register 23 High Word of Data Frame (PSI5_CH2_PMRL23)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x047C	PSI5 Message Register 24 Low Word of Data Frame (PSI5_CH2_PMRL24)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2158
0x0480	PSI5 Message Register 24 High Word of Data Frame (PSI5_CH2_PMRL24)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159

Table 1131. PSI5 memory map (continued)

Offset from PSI5 _BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x0484	PSI5 Message Register 25 Low Word of Data Frame (PSI5_CH2_PMRL25)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0488	PSI5 Message Register 25 High Word of Data Frame (PSI5_CH2_PMRL25)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x048C	PSI5 Message Register 26 Low Word of Data Frame (PSI5_CH2_PMRL26)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0490	PSI5 Message Register 26 High Word of Data Frame (PSI5_CH2_PMRL26)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x0494	PSI5 Message Register 27 Low Word of Data Frame (PSI5_CH2_PMRL27)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x0498	PSI5 Message Register 27 High Word of Data Frame (PSI5_CH2_PMRL27)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x049C	PSI5 Message Register 28 Low Word of Data Frame (PSI5_CH2_PMRL28)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x04A0	PSI5 Message Register 28 High Word of Data Frame (PSI5_CH2_PMRL28)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x04A4	PSI5 Message Register 29 Low Word of Data Frame (PSI5_CH2_PMRL29)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x04A8	PSI5 Message Register 29 High Word of Data Frame (PSI5_CH2_PMRL29)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x04AC	PSI5 Message Register 30 Low Word of Data Frame (PSI5_CH2_PMRL30)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158
0x04B0	PSI5 Message Register 30 High Word of Data Frame (PSI5_CH2_PMRL30)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2159
0x04B4	PSI5 Message Register 31 Low Word of Data Frame (PSI5_CH2_PMRL31)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf- Relv	on page 2158

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x04B8	PSI5 Message Register 31 High Word of Data Frame (PSI5_CH2_PMRL31)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2159
0x04BC	SMC Frame RX Register 1 (PSI5_CH2_SFR1)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x04C0	SMC Frame RX Register 2 (PSI5_CH2_SFR2)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x04C4	SMC Frame RX Register 3 (PSI5_CH2_SFR3)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x04C8	SMC Frame RX Register 4 (PSI5_CH2_SFR4)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x04CC	SMC Frame RX Register 5 (PSI5_CH2_SFR5)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x04D0	SMC Frame RX Register 6 (PSI5_CH2_SFR6)	R	R/W ⁽⁷⁾	R	0x0000_0000	Saf-Relv	on page 2161
0x04D4	New Data Status Register (PSI5_CH2_NDSR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2163
0x04D8	Over Write Status Register (PSI5_CH2_OWSR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2163
0x04DC	Error Status Register (PSI5_CH2_EISR)	R	R/W1C/ HC ⁽⁸⁾	R/W1C/ HC ⁽⁸⁾	0x0000_0000	No-Saf	on page 2164
0x04E0	Set New Data Status Register (PSI5_CH2_SNDSR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2165
0x04E4	Set Overwrite Status Register (PSI5_CH2_SOWSR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2165
0x04E8	Set Error Status Register (PSI5_CH2_SEISR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2166
0x04EC	Set SMC Error Status Register (PSI5_CH2_SSESR)	R	R/W1S	R/W1S	0x0000_0000	Saf-Relv	on page 2167
0x04F0	Sync Pulse Start Time Stamp Read Register (PSI5_CH2_STSRR)	R	R	R	0x0000_0000	Saf-Relv	on page 2168
0x04F4	Data Reception Time Stamp Read Register (PSI5_CH2_DTSRR)	R	R	R	0x0000_0000	Saf-Relv	on page 2169
0x04F8	Slot 1 Frame Configuration Register (PSI5_CH2_S1FCR)	R	R/W	R	0x0000_0010	Saf-Relv	on page 2169

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x04FC	Slot 2 Frame Configuration Register (PSI5_CH2_S2FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x0500	Slot 3 Frame Configuration Register (PSI5_CH2_S3FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x0504	Slot 4 Frame Configuration Register (PSI5_CH2_S4FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x0508	Slot 5 Frame Configuration Register (PSI5_CH2_S5FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x050C	Slot 6 Frame Configuration Register (PSI5_CH2_S6FCR)	R	R/W	R	0x0000_0010	Saf- Relv	on page 2169
0x0510	Slot 1 Start Boundary Register (PSI5_CH2_S1SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x0512	Slot 2 Start Boundary Register (PSI5_CH2_S2SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x0514	Slot 3 Start Boundary Register (PSI5_CH2_S3SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x0516	Slot 4 Start Boundary Register (PSI5_CH2_S4SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x0518	Slot 5 Start Boundary Register (PSI5_CH2_S5SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x051A	Slot 6 Start Boundary Register (PSI5_CH2_S6SBR)	R	R/W	R	0x0000	Saf- Relv	on page 2170
0x051C	Slot <i>n</i> End Boundary Register (PSI5_CH2_S <i>n</i> EBR)	R	R/W	R	0x0000_0000	Saf- Relv	on page 2171
0x0520	Data Output Block Configuration Register (PSI5_CH2_DOBCR)	R	R/W/ W1S ⁽⁹⁾	R/W1S ⁽⁹⁾	0x0000	Saf- Relv	on page 2172
0x0522	Manchester Decoder Disable Offset (PSI5_CH2_MDDIS_OFF)	R	R/W	R	0x0000	Saf- Relv	on page 2175
0x0524	Pulse Width for Data Bit Value 0 (PSI5_CH2_PW0D)	R	R/W	R	0x0000	Saf- Relv	on page 2176
0x0526	Pulse Width for Data Bit Value 1 (PSI5_CH2_PW1D)	R	R/W	R	0x0000	Saf- Relv	on page 2177
0x0528	Counter target Pulse Register (PSI5_CH2_CTPR)	R	R/W	R	0x0000	Saf- Relv	on page 2177
0x052A	Counter Initialize Pulse Register (PSI5_CH2_CIPR)	R	R/W	R	0x0000	Saf- Relv	on page 2178

Table 1131. PSI5 memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification ⁽³⁾	Location
		D	C	N			
0x052C	Data Preparation Register Low Word (PSI5_CH2_DPRL)	R	R/W ⁽¹⁰⁾	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2179
0x0530	Data Preparation Register High Word (PSI5_CH2_DPRH)	R	R	R	0x0000_0000	Saf-Relv	on page 2180
0x0534	Data Buffer Register Low Word (PSI5_CH2_DBRL)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2181
0x0538	Data Buffer Register (High Word (PSI5_CH2_DBRH)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2185
0x053C	Data Shift Register Low Word (PSI5_CH2_DSRL)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2187
0x0540	Data Shift Register High Word (PSI5_CH2_DSRH)	R	R	R/W ⁽¹⁰⁾	0x0000_0000	Saf-Relv	on page 2188
–0x0548- 0xFFFF	Reserved						

1. In this column, R/W = Read/Write, R = Read-only, W = Write-only, W1C = write-1-to-clear and W1S = “write-to-1-shot, always read as 0”, D = Disable mode, C = Config mode, N = Normal mode. All 32-bit registers are word/half word/byte addressable for write. Read is always word addressable. All 16-bit registers are half word/byte addressable for write. Read is always word addressable. Trying to access any register space of PSI5, other than those mentioned in the memory map, results in the generation of transfer error. Note that the generation of transfer error is confined to this scenario only.
2. In this column, the format of the reset value indicates the register width. Thus, reset values of the formats 0xHH, 0xHHHH, and 0xHHHH_HHHH, where “H” is a hexadecimal digit, indicate 8-bit, 16-bit, and 32-bit registers, respectively.
3. No-Saf = Not Safety Relevant, Saf-Relv = Safety Relevant.
4. The reset value is dependent on the parameter PSI5_PDIS_RST. When PSI5_PDIS_RST = 0 then the reset value is “0x0000” (PSI5 module is enabled after reset). When PSI5_PDIS_RST = 1 then the reset value is “0x0001” (PSI5 module is disabled after reset). For the value of parameter PSI5_PDIS_RST please refer to the Device Configuration File.
5. Not all the register bits are writable in this mode. Please see the details of the register in the Register Description section.
6. The bits can be cleared by hardware or by software using w1c. This is controlled by the “fast_clr_xxx” bit in the PSI5_PCCR. However not all bits of this register have this behavior. For details please see the register description section.
7. Writable ONLY in Config mode for test purposes. Otherwise updated by hardware.
8. The bits are set by hardware. Alternatively the software can also set these bits by writing to the corresponding SET registers. This provision should be used only for test purpose. Clearing of these bits is either by w1c or by hardware depending on the setting of the “fast_clr_xxx” bit in the PSI5_PCCR.
9. Few bits are writable to One Shot. For details please see the register details in the Register Description Section.
10. The bits of this register are writable ONLY when the corresponding DxR_RDY bit in the PSI5_GISR is equal to 1.

53.3.2 Register description

Unless specifically stated, the register bits are readable in all modes, but writable only in Config mode.

Note that, an access (read/write) to any register in the unimplemented register space will result in the generation of transfer error. All word locations are byte/half word/word accessible for write. Read is always a word access. Trying to do a word operation on 16-bit registers will NOT result in the generation of any transfer error. Trying to write to “read only” bits of implemented register space, will be ignored without the generation of any error. Trying to write to any register outside its intended mode of write (Normal/config) will be ignored without the generation of any transfer error.

For a summary of the register bit accesses in various device modes, please see [Section 53.3.2.42, Device modes and Register bit accesses](#)

53.3.2.1 Global Control Register (PSI5_GCR)

The Global Control Register (PSI5_GCR) provides control functions for all the PSI5 channels simultaneously.

Figure 1127. Global Control Register (PSI5_GCR)

Address: 0000H Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTC_GED	GLOBAL_DISABLE_REQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(1)

1. The reset value is dependent on the parameter PSI5_PDIS_RST. When PSI5_PDIS_RST = 0, then the reset value is “0” (PSI5 module is enabled after reset). When PSI5_PDIS_RST = 1 then the reset value is “1” (PSI5 module is disabled after reset). For the value of parameter PSI5_PDIS_RST please refer to the Device Configuration Chapter.

Table 1132. PSI5_GCR field descriptions

Field	Description
14 CTC_GED	Channel Target Counter (CTC) Global Enable/Disable. 0 All the CTC counters having PCCR[CTC_GED_SEL] = 1 are reset. 1 Simultaneously starts the CTC of all the channels that have PCCR[CTC_GED_SEL] = 1. <i>Note: This bit is writable in all modes.</i>
15 GLOBAL_DISABLE_REQ	PSI5 Receive Global disable request. When set, all PSI5 channels enter Disable mode (from individual Normal or Config modes) regardless of the settings of the PCCR[PSI5_CH_CONFIG] and PCCR[PSI5_CH_EN] bits. When cleared, each PSI5 channel enters Config mode or Normal mode based on the setting of the PCCR[PSI5_CH_CONFIG] bit if PCCR[PSI5_CH_CONFIG] is set for that channel; otherwise, that particular channel continues to stay in Disable mode. The reset value is dependent on the parameter PSI5_PDIS_RST. When PSI5_PDIS_RST = 0 then the reset value is “0” (PSI5 module is enabled after reset). When PSI5_PDIS_RST = 1 then the reset value is “1” (PSI5 module is disabled after reset). For the value of parameter PSI5_PDIS_RST please refer to the Device Configuration Chapter. This bit is writable in all the modes. 0 PSI5 channel(s) enters Normal or Config mode, depending on the value of the PCCR[PSI5_CH_CONFIG] bit. 1 All PSI5 Channels are in Disable mode. <i>Note: When GLOBAL_DISABLE_REQ == 1 then all the clock enables (shown in Figure 1192) from the IP go to a 0 (inactive) state.</i>

53.3.2.2 PSI5 Channel Control Register (PSI5_PCCR)

Figure 1128 shows the PSI5 Channel Control Register.

Figure 1128. PSI5 Channel Control Register (PSI5_PCCR)

Address: Base + 0x0008 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CTC_GED_SEL	CTC_ED	0	MEM_DEPTH					0	0	0	ERROR_SELECT[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	GTM_RESET_ASYNC_EN	0	0	0	DEBUG_EN	DEBUG_FREEZE_CTRL	SP_TS_CLK_SEL	0	0	FAST_CLR_SMC	FAST_CLR_PSI5	BIT_RATE	MODE	PSI5_CH_CONFIG	PSI5_CH_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1133. PSI5_PCCR field descriptions

Field	Description
0 CTC_GED_SEL	Channel Target Counter, Global Enable/Disable Select. This bit selects whether the CTC should be enabled/disabled by CTC_GED bit (in GCR) or the CTC_ED bit. Enabling by the CTC_GED bit is used when staggering of the channels is required. 0 CTC enabled/disabled by CTC_ED. 1 CTC enabled/disabled by CTC_GED.
1 CTC_ED	Channel Target Counter Enable/Disable. This bit is used to control the CTC locally from the channel in case the CTC_GED_SEL = 0. 0The CTC counters is disabled and reset. 1The CTC counter is enabled and start counting. <i>Note: This bit is writable in Normal mode and the Config mode.</i>
3–7 MEM_DEPTH	Can be programmed from 0 to 31 and denotes the size of the memory that should be used for storing the PSI5 messages. Area above the MEM_DEPTH is treated as being unavailable for message storage. MEM_DEPTH also denotes the number of RAM registers available for the CPU access. Note that the same memory can be accessed sequentially (though the DMA) or randomly anytime by the CPU. 0Indicates that only 1 location of the memory is available for storage 31Indicates that all the 32 locations of the memory are available for storage. <i>Note:</i> – Depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR, the MEM_DEPTH memory size may be accessed by CPU only (interrupts are generated) or CPU and DMA (through the dedicated DMA popup registers). – DMA_PM_DS_CONFIG = conf1 then DMA requests are disabled and only the interrupts can be used for data transfer. – DMA_PM_DS_CONFIG = conf2,conf3,conf4 then DMA requests are enabled. The interrupts should be disabled by writing a 0 in the corresponding IE bits. – For more details please refer to description of DMA_PM_DS_CONFIG in the DCRs. – The data in the memory area defined by the MEM_DEPTH parameter is always filled in a sequential fashion as in a FIFO, but can be read sequentially or randomly. There can be no slot-memory location correspondence due to the T bit errors in the Manchester decoder. – The memory area above the configured MEM_DEPTH size will always be read as 0. – Irrespective of the configuration, the data can always be randomly read by the CPU from the PMRL/PMRH register.
11 ERROR_SELECT[4]	The ERROR_SELECT bitfield indicates which of the C, E, EM, T, or F error conditions generate an interrupt when the EICR Interrupt Enable bit is set. The individual bits are mapped as {C,E,EM,T,F}. The corresponding error is also latched in the PSI5_EISR. Default: ERROR_SELECT = 11111: Any of C,E,EM,T,or F generate an interrupt. C error CRC Error (C) Bit 20: ERROR_SELECT[4] or C_INT_SEL 0 The C bit does not generate an interrupt. 1 The C bit generates an interrupt.

Table 1133. PSI5_PCCR field descriptions (continued)

Field	Description
12 ERROR_SELECT[3]	Error_Select[3] or E_INT_SEL E error Electrical Error (E) 0 The E bit does not generate an interrupt. 1 The E bit generates an interrupt.
13 ERROR_SELECT[2]	Error_Select[2] or EM_INT_SEL 0 The EM bit does not generate an interrupt. 1 The EM bit generates an interrupt.
14 ERROR_SELECT[1]	Error_Select[1] or T_INT_SEL 0 The T bit does not generate an interrupt. 1 The T bit generates an interrupt.
15 ERROR_SELECT[0]	Error_Select[0] or F_INT_SEL 0 The F bit does not generate an interrupt. 1 The F bit generates an interrupt.
17 GTM_RESET_ASYNC_EN	GTM_RESET_ASYNC_EN 0 Both the assertion and the deassertion of the GTM reset are treated synchronously. Inside the IP the GTM reset is synchronized on the “ipg_clk_psi5”. This can delay the impact of the GTM reset by 1 to 2 system clock(ipg_clk_psi5). Please refer to Figure 1192 for the definition of “ipg_clk_psi5”. 1 Assertion of the GTM reset is treated asynchronously but deassertion is synchronized on the ipg_clk_psi5.
21 DEBUG_EN	This bit allows/prevents the IP from entering the debug mode, whenever the debugger is connected and a breakpoint is encountered by the debugger: 0 The IP never enters the debug mode, even if the debugger is connected and a breakpoint is encountered. It keeps on working normally, as in the functional mode. 1 Whenever the debugger is connected and a breakpoint is encountered, the IP enters the debug mode. In the Debug mode, the DMA read pointers and other status registers that normally get affected by the READ operation remain unaffected.
22 DEBUG_FREEZE_CTRL	0 When the IP enters the Debug mode then it goes into the “Open” mode, i.e., the core of the IP keeps on working normally. In this case, there can be a difference between the value of a particular status bit, which is there at the time Debug mode is entered and the time at which the debugger reads it, since some upcoming events might change the status. The hardware clearing, of course, remains disabled. <i>Note: In the above configuration, the status of registers read through the debugger may be different when entering debug mode and when actually accessing registers at some later point of time.</i> 1 When the IP enters the “Debug mode” then it goes into the “Freeze Mode”. Note that whenever there is a request to enter the freeze mode then the IP enters this mode, ONLY when the Manchester decoder and the Sync Pulse Generator are stopped coherently. This means that before entering the “freeze mode”, if there is any pending reception that is ongoing, then the same is completed first. Simultaneously the “sdout” line of the IP should be at a low level. Once these two conditions are satisfied, the IP enters the Debug Freeze mode. The Time Stamp counters are also halted. In the above configuration, incoming packets could get lost while the Debug mode is active.

Table 1133. PSI5_PCCR field descriptions (continued)

Field	Description
23 SP_TS_CLK _SEL	This bit controls which clock goes to the Sync Pulse and Time Stamp Generator Unit. 0 The 1 MHz clock generated from the CGM goes for clocking the Time Stamp Unit and sync pulse generator. 1 The clock generated from the GTM goes for clocking the Time Stamp Unit and Sync Pulse Generator. <i>Note: The above selection affects ONLY the clock for the Time Stamp Unit and Sync Pulse Generator. The 1 MHz clock to the rest of the logic like the Manchester decoder or the Slot Boundary Counters is still the 1 MHz from the CGM.</i> <i>Note: The clock selected using this bit is called SP_TS_CLK.</i>
26 FAST_CLR _SMC	This bit controls the clearing mechanism of the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits in the GISR. 0 Fast clearing is disabled (clear when written to 1). 1 Fast clearing is enabled. <i>Note: This bit is effective only when DCR[DMA_EN_SF] = 1. When FAST_CLR_SMC = 1 then the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits are automatically cleared upon reading the DSFR message register, as per the slot[x] message read in from the DSFR. When reading directly from the SFR registers, this bit does not affect auto clearing.</i> <i>If FAST_CLR_SMC = 0 then the above bits have to be cleared by a w1c.</i>

Table 1133. PSI5_PCCR field descriptions (continued)

Field	Description
27 FAST_CLR_PSI5	This bit control the clearing mechanism of bits in NDSR (in conf2, conf3 and conf4 modes), EISR (in conf2, conf3 and conf4 modes), and OWSR (in conf2 and conf3 modes). 1 Fast Clearing Enabled 0 Fast Clearing Disables (clear when written to 1) <i>Note: This bit is effective only when DMA_PM_DS_CONFIG = conf2, conf3, or conf4 else this bit has no effect and only software clearing is available. Further, though the OWSR is not directly read by the DMA as a single register, it is cleared in specific modes (when FAST_CLR_PSI5 == 1) as can be seen below.</i> Following are the details in the various modes of the DCR[DMA_PM_DS_CONFIG] bits: – In conf1 mode: This bit has no effect. Reading the DDSR and the DPMR returns a 0. – In conf2 mode: In this case the diagnostic bits (EISR, NDSR) are cleared ONLY when the diagnostic registers appended with the corresponding PSI5 messages are completely read using the DPMR. During the intermediate operation (when the PSI5 message has been read but the diagnostic bits are still pending), the PSI5 message read from the DPMR will not clear these bits. Both of these registers are simultaneously cleared when the corresponding DMA operation is over. However, only those bits are cleared, the corresponding messages of which form the part of the DMA request being serviced. Note that the OWSR bits are cleared upon the corresponding PSI5 message read from the DPMR. Reading the DDSR returns a 0 in this mode. – In conf3 mode: The PSI5 diagnostic register bits (EISR,NDSR) and OWSR are automatically cleared when the corresponding PSI5 message is read from the DPMR. The specific register bits of EISR,NDSR and OWSR, corresponding to each message, are cleared sequentially one by one as the corresponding message is being read by the DMA. Note that the three bits (each of EISR, NDSR, and OWSR) corresponding to a specific location are always cleared simultaneously. Reading the DDSR returns a 0. When reading from the PMRH/PMRL registers, this bit does not have any effect on fast clearing. – In conf4 mode: In this case the diagnostic bits (EISR, NDSR) are cleared as soon as these registers are read through the DDSR. These registers are cleared simultaneously in one go, as soon as the corresponding DMA request has been serviced. Only those bits of these registers are cleared, the messages of which form a part of the DMA request. Note that in this mode the reading of the PSI5 messages from the DMA is not available as the request is being used for reading the diagnostic registers. Reading the DPMR returns a 0. The PSI5 messages can be read only from the PMRH/PMRL register. Hence the OWSR has to be cleared only by w1c. – In all the above modes, when reading from the PMRH/PMRL registers, the FAST_CLR_PSI5 does not have any effect on fast clearing. – Further, In all the above cases if FAST_CLR_PSI5 = 0 then the specific diagnostic bits and the OWSR have to be cleared by w1c.
28 BIT_RATE	This bit selects the receive message bit rate (T bit) for this particular PSI5 Channel, that is, it selects one of the two driven clocks (4 MHz or 6.048 MHz) in CGM. 0 125 Kbit/s bit rate selected (4 MHz clock). 1 189 Kbit/s bit rate selected (6.048 MHz clock).
29 MODE	This bit selects the operating modes. 0 Asynchronous operating mode (Integrated sync pulse generator module is switched off, only RX active). Note that in this mode the sync pulses from the internal sync pulse generator or the GTM are not allowed to be propagated to the output path, which always remains at logic 0. There is no T-bit error in the Asynchronous mode. 1 Synchronous operating modes (Integrated sync pulse generator module is switched on, both TX and RX subblocks active).

Table 1133. PSI5_PCCR field descriptions (continued)

Field	Description
30 PSI5_CH_CONFIG	PSI5 Channel Config mode request. If this bit is set (and PCCR[PSI5_CH_EN] = '1') this particular psi5 channel enters Configuration mode from Disable mode. Default value is 0. User software must program all PSI5 channel parameters before clearing this bit. Once cleared, channel parameters in registers are locked for writing. The values set cannot be changed until this bit is set again after entering the Disable mode 1 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') - PSI5 Channel enters Config mode. Various configuration registers can be programmed in this mode. The only registers NOT writable in the config mode but writable in the Normal mode are the output registers PSI5_DBR and the PSI5_DSR. 0 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') PSI5 Channel enters Normal mode. Channel parameter registers are locked for writing. This bit can be written in all the modes but writing a "0" in the Normal mode will not change the operation mode of the device. <i>Note: This bit can be written in all modes, but writing a 0 in the Normal mode does not change the operation mode of the device. For more details please see Figure and Table 1129. Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written.</i>
31 PSI5_CH_EN	PSI5 Channel Enable. If this bit is cleared this particular psi5 channel continues to stay in Disable mode even if GCR[GLOBAL_DISABLE_REQ] = 0. If set, it enters Configuration mode from Disable mode (based on PCCR[PSI5_CH_CONFIG]). Default value is 0. All state machines are disabled. This bit can be written in all modes. 0 PSI5 channel continues in Disable mode. 1 PSI5 channel enabled to enter Config/Normal mode. <i>Note: When PSI5_CH_EN == 0 then the clock enable for that particular channel that goes from the IP, goes to "0" (inactive state). " "The enable for the GTM Clock is not controlled by the IP and should be controlled externally. Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written.</i>

53.3.2.3 DMA Control Register (PSI5_DCR)

[Figure 1129](#) shows the DMA Control Register.

Figure 1129. DMA Control Register (PSI5_DCR)

Address: Base + 0x000C + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	DMA_PM_DS_WM[4:0]					0	0	0	0	0	IE_DMA _TF_SF	IE_DMA _TF_PM_DS	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	IE_DMA_PM _DS_FIFO_FULL	IE_DMA_SFUF	0	IE_DMA _PM_DS_UF	0	0	0	0	0	DMA_EN_SF	DMA_PM _DS_CONFIG	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1134. PSI5_DCR field descriptions

Field	Description
3–7 DMA_PM_DS_WM[4:0]	0 to 31Valid water mark levels. Value fixed by user software. Since each location has two 32-bit registers hence a total of sixty-four 32-bit words are present. The default and minimum value of dma_pm_ds_wm is '0', i.e., 1 message/Diagnostic bit of 1 location. The maximum value of dma_pm_ds_wm is 31, i.e., the actual watermark = 32 RAM locations. <i>Note:</i> – These bits are ineffective if DMA_PM_DS_CONFIG = conf1. – These bits indicates the number of PSI5 message to be stored in FIFO / number of unread diagnostic bits in the diagnostic registers with a correspondence to the messages in the FIFO, before the corresponding DMA request is asserted. The watermark definition changes depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR as mentioned below. – When DMA_PM_DS_CONFIG = conf2 or conf3 then the watermark refers to the number of new unread PSI5 messages that are stored in the FIFO before a DMA request is asserted. – In conf2 mode of DMA_PM_DS_CONFIG, the number of words to be transferred, set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value + 2 words (for the two 32-bit diagnostic registers). – In conf3, the number of words to be transferred set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value. – When DMA_PM_DS_CONFIG = conf4 then the watermark refers to the number of new unread diagnostic bits in the diagnostic register. In this configuration the PSI5 messages can only be read by the CPU, since the DMA request would cater to reading the diagnostic bits only. – In conf4, the number of words transferred in each DMA request = 2 x 32-bit words (for the two 32-bit diagnostic registers). The water mark however refers to the number of new unread diagnostic bits pertaining to each PSI5 message in the FIFO.
13 IE_DMA_TF_SF	Enable for interrupt, which is generated when DMA transfer finishes for DMA SMC Frame register. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
14 IE_DMA_TF_PM_DS	Enable for Interrupt, which is generated when DMA transfer finishes for PSI5 messages/DMA Diagnostic Status register depending on the DMA_PM_DS_CONFIG bits of the DCR. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
20 IE_DMA_PM_DS_FIFO_FULL	This bit is effective only when the DMA_PM_DS_CONFIG = conf2,conf3 or conf4. It enables the FIFO FULL condition generation interrupt. For the cases that generate these FIFO FULL conditions, please refer to the description of IS_DMA_PM_DS_FIFO_FULL register bit of the DSR. <i>Note: This bit is writable in Config and Normal modes.</i>
21 IE_DMA_SFUF	Enables interrupt when there is underflow in DMA SMC Frame register when DMA is enabled. It is set when the DSFR is read without a valid DMA request, i.e., it is empty. Default value is 0. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>

Table 1134. PSI5_DCR field descriptions (continued)

Field	Description
23 IE_DMA_PM_DS_UF	Enables interrupt when there is underflow in DMA PSI5 message register when DMA is enabled. Default value is 0. For the details as to when this underflow occurs, please refer to the details of IS_DMA_PM_DS_UF bit details of the DSR. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
29 DMA_EN_SF	Enable DMA request for SMC Frame Data 0 DMA for SMC frame is disabled. The six dedicated 1-1 slot corresponding SMC registers can only be read by the CPU through the SFR. Reading the DSFRs returns 0. 1 DMA for SMC frame is enabled. The six dedicated slot corresponding SMC registers can be read by reading the DSFR successively. In the DMA mode the six registers are read in a round robin fashion as explained in the DSFR details. Reading directly through the SFR registers is still available.
30–31 DMA_PM_DS_CONFIG	These bits define how the PSI5 messages are stored in the MEM_DEPTH memory area and the associated diagnostic bits (EISR, NDSR) are transferred. 00 conf1 The DMA request is disabled. In this mode, only the interrupts can be used for indicating that data transfer is required. 01 conf2 After transferring the “dma_pm_ds_wm” number of PSI5 messages, there will be two additional 32-bit transfers in which the NDSR and the EISR registers are sequentially transferred through the DPMR registers by using DMA. 10 conf3 “dma_pm_ds_wm” number of PSI5 messages is transferred in each DMA request. The DMA read is through the DPMR registers. 11 conf4 Only the diagnostic bits are transferred through the DDSR. The DMA read is through the DDSR registers. The DPMR, DDSR, and DSFR registers should be used for DMA access and the PMRH/PMRL, NDSR, EISR, and OWSR registers should be used for CPU access. <i>Note: For more details please see the description of the DCR[DMA_PM_DS_WM] bits.</i> <i>When DMA_PM_DS_CONFIG bits = conf2 or conf3 and if the CPU and the DMA are simultaneously used for reading the PSI5 messages, then in the course of reading the messages it is possible that there can be an overflow, though some intermediate locations, which have been read by the CPU, are empty. This will occur because the DMA access is made independent of the CPU access. The DMA logic would never know that the CPU has already read a message that was scheduled to be read by the DMA.</i> <i>Whenever the DMA is used for reading the PSI5 message (conf2/conf3) modes then though the CPU can read the messages in parallel through the PMRH/PMRL registers, still for the purpose of setting the Overwrite/Overflow bits only a read from the DMA through the popup registers (DPMR) would be treated as valid read. This will also be true for the SMC messages when accessed simultaneously by the CPU/DMA.</i>

Figure 1130 shows the DMA Status Register.

53.3.2.4 DMA Status Register (PSI5_DSR)

Figure 1130. DMA Status Register (PSI5_DSR)

Address: Base + 0x0010 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	IS_DMA_ TF_SF	IS_DMA_ TF_PM_DS	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	IS_DMA_PM_ DS_FIFO_FULL	IS_DMA_SFUF	0	IS_DMA_PM_ DS_UF	0	0	0	0	0	0	0	0
W					w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1135. PSI5_DSR field descriptions

Field	Description
13 IS_DMA_TF_SF	This flag is set when DMA transfer finishes for DMA SMC Frame register.
14 IS_DMA_TF_PM_DS	This flag is set when DMA transfer finishes. For the various configurations of this request please refer to the descriptions of the DMA_PM_DS_CONFIG bits in the DCR. This flag is cleared by w1c.

Table 1135. PSI5_DSR field descriptions (continued)

Field	Description
20 IS_DMA_PM_DS_FIFO_FULL	Interrupt Status when there is FIFO FULL corresponding to the DMA request ipd_psi5_dma_req_pm_ds. The FIFO FULL scenarios are detailed below with respect to the various settings of the DMA_PM_DS_CONFIG bit. conf1: FIFO FULL disabled conf2 and conf3: FIFO FULL occurs when all the memory locations become full in the PSI5 message FIFO. It indicates that without any read, any new upcoming message will now overwrite the existing messages and set the overwrite bits. This bit is cleared either by w1c or upon a PSI5 message read depending on the FAST_CLR_PSI5 bit. conf4: FIFO FULL occurs when there is a “REGISTER FULL” in the Diagnostic registers. FIFO FULL here means that the diagnostic registers contain 32 unread diagnostic bits. A new message starts changing the status of the diagnostic bits. In this configuration it is necessary to read the diagnostic bits through the DMA using the DDSR in order to prevent FIFO FULL. Of course the diagnostic registers NDSR/EISR can be read any time by the CPU, but to prevent FIFO FULL they HAVE to be read through the DDSR registers only. Also in this mode the PSI5 messages DONT contribute to the FIFO FULL. The PSI5 messages have to be read by the CPU using the PMRH/PMRL registers in order to prevent the overwrite bits from being set in the OWSR. In this mode the OWSR bits are NOT automatically cleared upon the PSI5 message read since the PSI5 message has to be read by the PMRH/PMRL register and NOT though the DPMR. The OWSR bits have to be cleared by a w1c. Reading the DPMR returns a 0 in this mode. This bit is cleared by a w1c. 0 No FIFO full. 1 FIFO full has occurred
21 IS_DMA_SFUF	SMC Frame DMA underflow: This happens when the DSFR has been read without a proper DMA request being asserted. The DSFR is empty and it is read. This bit is cleared by a w1c. 0 No underflow has occurred. 1 Underflow has occurred.
23 IS_DMA_PM_DS_UF	Depending on the DMA_PM_DS_CONFIG bits following is the underflow conditions: Default value is 0. conf1: underflow disabled conf2 and conf3: Underflow happens when the software reads the PSI5 message FIFO though the DPMR, beyond the available messages (empty area). conf4: Underflow happens when the diagnostic registers are read through the DDSR without a valid DMA request and the DDSR contains no new data. 0 No underflow has occurred. 1 Underflow has occurred. The bits are cleared by a w1c.

53.3.2.5 General Interrupt Control Register (PSI5_GICR)

The General Interrupt Control Register (PSI5_GICR) contains the bits for controlling the SMC message related interrupts, interrupts related to ECU-to-sensor communication, and interrupts to the Time Stamp.

Figure 1131. General Interrupt Control Register (PSI5_GICR)

Address: Base + 0x0014 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	IE_CESM[6:1]						IE_STS	IE_DTS	IE_DSROW	IE_BROW	IE_PROW	IE_DSRR	IE_BRR	IE_PRR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	IE_OWSM[6:1]						0	0	IE_NVSM[6:1]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1136. PSI5_GICR field descriptions

Field	Description
2–7 IE_CESM[6:1]	Interrupt request when received SMC frame in corresponding slot has CRC failure (CRC recalculation on SMC). 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: These bits are writable in Config and Normal modes.</i>
8 IE_STS	Interrupt request enabled when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
9 IE_DTS	Interrupt request enabled when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
10 IE_DSROW	Interrupt request enabled when system tries to overwrite on Data Shift register when it is not ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
11 IE_BROW	Interrupt request enabled when system tries to overwrite on buffer register when it is not ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
12 IE_PROW	Interrupt request enabled when system tries to overwrite on Preparation register when it is not ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>

Table 1136. PSI5_GICR field descriptions (continued)

Field	Description
13 IE_DSRR	Interrupt request enabled when Data Shift Register is ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
14 IE_BRR	Interrupt request enabled when buffer Register is ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
15 IE_PRR	Interrupt request enabled when Preparation Register ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: This bit is writable in Config and Normal modes.</i>
18–23 IE_OWSM[6:1]	Interrupt request enable for the SMC message Overwrite bits. The interrupt for SMC message overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten the overwrite bit gets set. <i>Note: These bits are writable in Config and Normal modes.</i>
26–31 IE_NVSM[6:1]	Interrupt request enabled when any new valid SMC message (fault free) is received in SFR[i]{i:0 to 5}, i.e., during slot1 to slot6. This interrupt is generated only when corresponding IS_NVSMS[5:0] is set. For each bit position, the corresponding request is enabled as follows: 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: These bits are writable in Config and Normal modes.</i>

53.3.2.6 New Data Interrupt Control Register (PSI5_NDICR)

The New Data Interrupt Control Register (PSI5_NDICR) contains the interrupt enable bits related to a new PSI5 message arrival, in the corresponding PSI5 message buffer location.

Figure 1132. New Data Interrupt Control Register (PSI5_NDICR)

Address: Base + 0x0018 + (0x1C0 × channel)

Access: User read/write

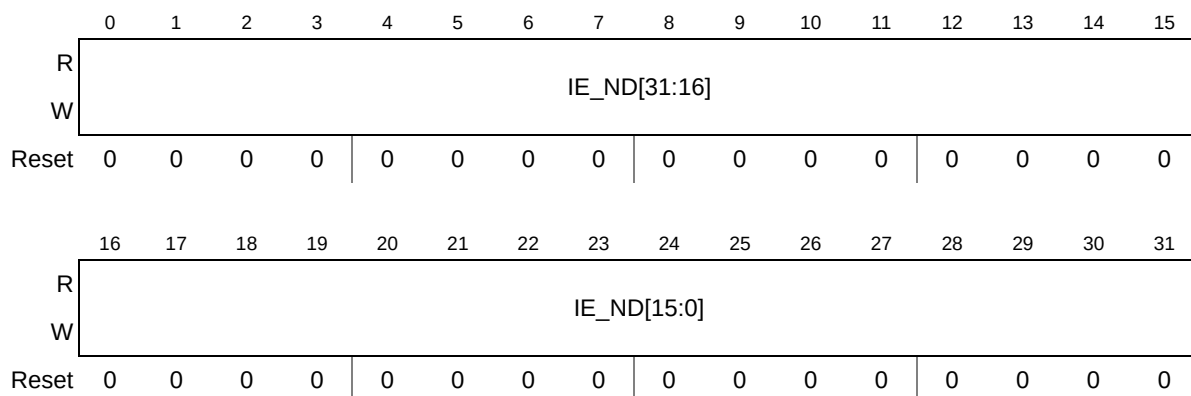


Table 1137. PSI5_NDICR field descriptions

Field	Description
0–31 IE_ND[31:0]	Interrupt request enabled when any new message (fault-free/with fault) is received in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when corresponding RAM buffer Ready with new data. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: These bits are writable in Config and Normal modes.</i>

Note: Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

53.3.2.7 Overwrite Interrupt Control Register (PSI5_OWICR)

The Overwrite Interrupt Control Register (PSI5_OWICR) is related to the generation of the overwrite interrupts. The overwrite interrupt is generated when the unread PSI5 message in a particular RAM location is overwritten by another PSI5 message.

Figure 1133. Overwrite Interrupt Control Register (PSI5_OWICR)

Address: Base + 0x001C + (0x1C0 × channel)

Access: User read/write

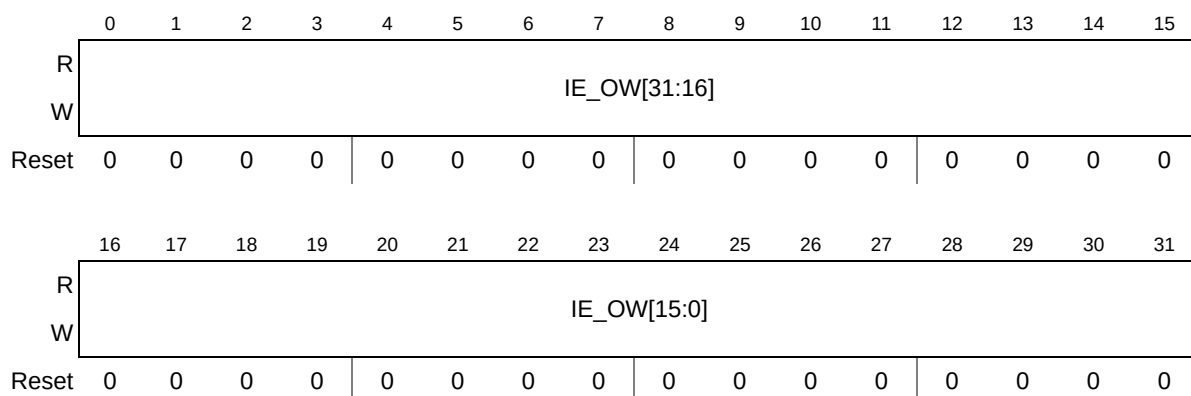


Table 1138. PSI5_OWICR field descriptions

Field	Description
0–31 IE_OW[31:0]	Interrupt request enabled when any new message overwrites the old unread PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt generated only when corresponding RAM buffer Ready with new data. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: These bits are writable in Config and Normal modes.</i>

Note: Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

53.3.2.8 Error Interrupt Control Register (PSI5_EICR)

[Figure 1134](#) shows the Error Interrupt Control Register.

Figure 1134. Error Interrupt Control Register (PSI5_EICR)

Address: Base + 0x0020 + (0x1C0 × channel) Access: User read/write

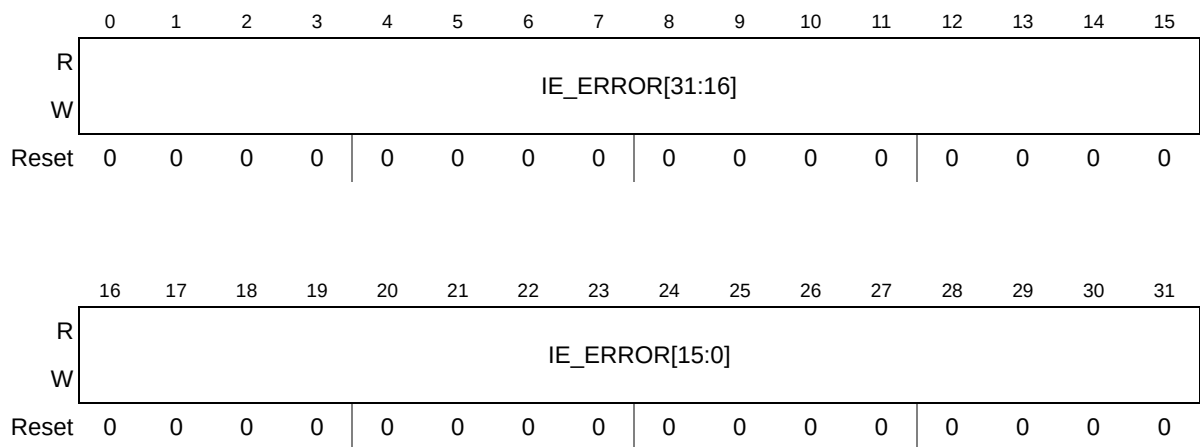


Table 1139. PSI5_EICR field descriptions

Field	Description
0–31 IE_ERROR [31:0]	Interrupt request enabled when any/all of the error conditions C, E, EM, T, or F is observed in a PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31]. This interrupt is generated only when the corresponding RAM buffer is Ready with new data. Which error condition (any of C, E, EM, T, or F) generates the interrupt is selectable by the PCCR[ERROR_SELECT] field. If all are selected, then which of C,E,EM,T or F is set can be isolated by reading the corresponding received message. 0 Interrupt is disabled. 1 Interrupt is enabled. <i>Note: These bits are writable in Config and Normal modes.</i>

Note: Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

53.3.2.9 General Interrupt Status Register (PSI5_GISR)

The General Interrupt Status Register (PSI5_GISR) contains the status bits related to SMC messages, ECU-to-sensor communication, and the Time Stamp.

Figure 1135. General Interrupt Status Register (PSI5_GISR)

Address: Base + 0x0024 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	IS_CESM[6:1]						IS_STS	IS_DTS	IS_DSROW	IS_BROW	IS_PROW	DSR_RDY	DBR_RDY	DPR_RDY
W			w1c						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	IS_OWSM[6:1]						0	0	IS_NVSM[6:1]					
W			w1c or clear on read of corresponding SMC message								w1c or clear on read of corresponding SMC message					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1140. PSI5_GISR field descriptions

Field	Description
2–7 IS_CESM[6:1]	These flags are set when the received SMC frame in the corresponding slot has a CRC failure (CRC recalculation on SMC). These bits are automatically cleared upon message read, when the FAST_CLR_SMC = 1 and the DMA_EN_SF = 1.
8 IS_STS	This Interrupt flag is set when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR.
9 IS_DTS	This Interrupt flag is set when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR.
10 IS_DSROW	This flag is set when the system tries to overwrite on Data Shift register when it is not ready to accept new data.
11 IS_BROW	This flag is set when the system tries to overwrite on buffer register when it is not ready to accept new data.
12 IS_PROW	This flag is set when the system tries to overwrite on Preparation register when it is not ready to accept new data.
13 DSR_RDY ⁽¹⁾	This bit acts both as a status and a control bit. Status bit Action: When “1” it indicates that the Data Shift register is ready to receive new data from the system bus side or the Data Buffer register. When “0” it indicates that there is a ongoing command transmission of the data in DSR, or the CPU has done a “w1c” to this bit leading to the new command transmission. In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_DSROW] bit gets set. A write to PSI5_DOBCCR[DSR_RST] will abort the current command transmission and make the DSR_RDY as “1”. Control Bit Action: When a “w1c” is done to this register bit, and PSI5_DOBCCR[SW_READY] == 1, then it indicates to the IP that the data in DSR is valid and is to be used for command shifting. This procedure is used when the data is written by the CPU to DSR directly skipping the DBR. For detailed action of this bit please refer to Section 53.4.3.4, Data transmission.

Table 1140. PSI5_GISR field descriptions (continued)

Field	Description
14 DBR_RDY ⁽²⁾	This bit acts both as a status and a control bit. Status bit Action: When “1” it indicates that the Data Buffer register is ready to receive new data from the system bus side or the Data Preparation register. When “0” it indicates that there is a ongoing shift of the data from the DPR to the DBR, or the CPU has done a “w1c” to this bit. In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_BROW] bit gets set. A write to PSI5_DOBCCR[DBR_RST] will abort the current data transaction in DBR and make the DBR_RDY as “1”. Control Bit Action: When a “w1c” is done to this register bit, then it indicates to the IP that the data in DBR is valid and is to be used for shifting to DSR. This procedure is used when the data is written by the CPU to DBR directly skipping the DPR. For detailed action of this bit please refer to Section 53.4.3.4, Data transmission.
15 DPR_RDY ⁽³⁾	This bit acts both as a status and a control bit. Status bit Action: When “1” it indicates that the Data Preparation register is ready to receive new data from the system bus side. When “0” it indicates that there is a ongoing processing of the data in DPR or there is a data shift ongoing from the DPR to the DBR, or the CPU has done a “w1c” to this bit. In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_PROW] bit gets set. When PSI5_DOBCCR[CMD_TYPE] == “7” then this bit remains at value “0”. Control Bit Action: When a “w1c” is done to this register bit, then it indicates to the IP that the data in DPR is valid and is to be used for processing and shifting to DBR. For detailed action of this bit please refer to Section 53.4.3.4, Data transmission.
18–23 IS_OWSM[6:1]	Status for the SMC message overwrite bits. Each of the 6 locations correspond to the SMC message pertaining to each SFR[x] register, x = 1 to 6. These bits are automatically cleared upon message read when the FAST_CLR_SMC = 1 and the DMA_EN_SF = 1.
26–31 IS_NVSM[6:1]	These flags are set when corresponding SFR[i] receives and is ready (to read) with new valid SMC message. These bits are automatically cleared upon message read, when the FAST_CLR_SMC = 1 and the DMA_EN_SF = 1.

1. For these register bits individual “bit setting” commands should not be used. Rather, the normal register read/write commands should be used.
2. For these register bits individual “bit setting” commands should not be used. Rather, the normal register read/write commands should be used.
3. For these register bits individual “bit setting” commands should not be used. Rather, the normal register read/write commands should be used.

53.3.2.10 DMA PSI5 Message Register (PSI5_DPMR)

[Figure 1136](#) shows the DMA PSI5 Message Register.

Figure 1136. DMA PSI5 Message Register (PSI5_DPMR)

Address: Base + 0x0028 + (0x1C0 × channel)

Access: User read/write

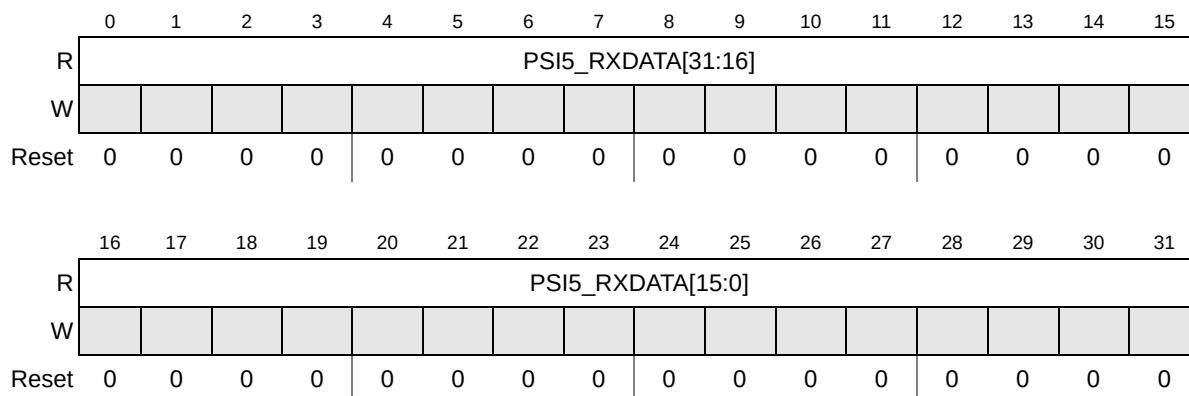


Table 1141. PSI5_DPMR field descriptions

Field	Description
0–31 PSI5_RXDATA [31:0]	PSI5_RXDATA[31:0] This popup register contain the PMRL[31:0] data followed by the PMRH[31:0] data for each of the FIFO locations, when the FIFO is read by the DMA. It would require two 32-bit IPS access to read one PSI5 Message. For each of the locations, the DATA is pushed sequentially until all the data corresponding to the programmed watermark is complete, after which the DMA request is deasserted. The depth of the FIFO is 32x64. This popup register can be used differently based on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. When DMA_PM_DS_CONFIG=conf2 then it pops the PSI5 message, followed by NDSR, followed by EISR. When DMA_PM_DS_CONFIG= conf3, it pops only the PSI5 messages. When DMA_PM_DS_CONFIG=conf4, it pops the NDSR followed by EISR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the Section 53.3.2.2, PSI5 Channel Control Register (PSI5_PCCR), Section 53.3.2.3, DMA Control Register (PSI5_DCR), and Section 53.3.2.4, DMA Status Register (PSI5_DSR).

53.3.2.11 DMA SMC Frame Register (PSI5_DSFR)

Figure 1137 shows the DMA SMC Frame Register.

Figure 1137. DMA SMC Frame Register (PSI5_DSFR)

Address: Base + 0x002C + (0x1C0 × channel)

Access: User read/write

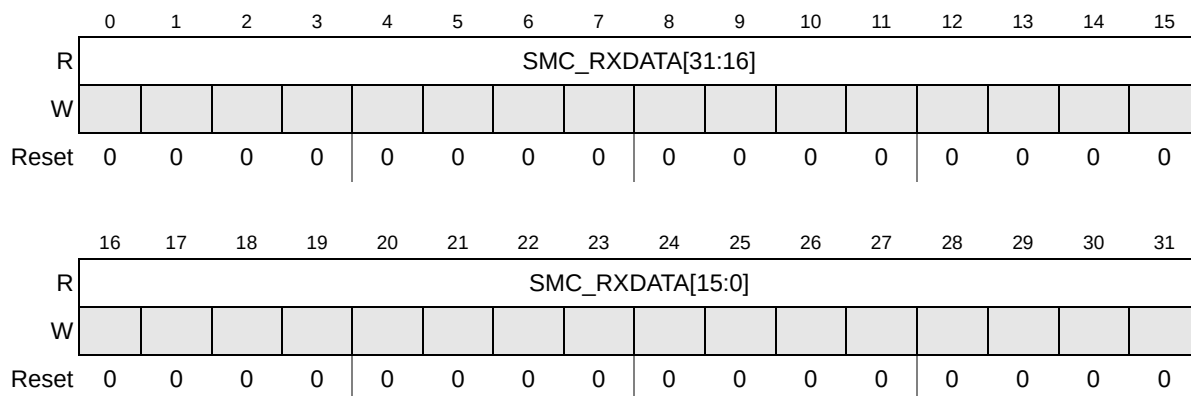


Table 1142. PSI5_DSFR field descriptions

Field	Description
0–31 SMC_RXDATA[31:0]	When the DMA_EN_SF = 1, then the six SFR registers are searched in a round robin fashion for the reception of the complete SMC data. The DMA request is asserted as soon as the first encountered SFR has a complete SMC frame. This request remains asserted until all the registers that have pending data, have transferred the data through the DMA. Each other SFR is a 32-bit register. This register is a reflection of the location pointed to by the DMA read pointer.

53.3.2.12 DMA Diagnostic Status Register (PSI5_DDSDR)

Figure 1138 show the DMA Diagnostic Status Register.

Figure 1138. DMA Diagnostic Status Register (PSI5_DDSDR)

Address: Base + 0x0030 + (0x1C0 × channel)

Access: User read/write

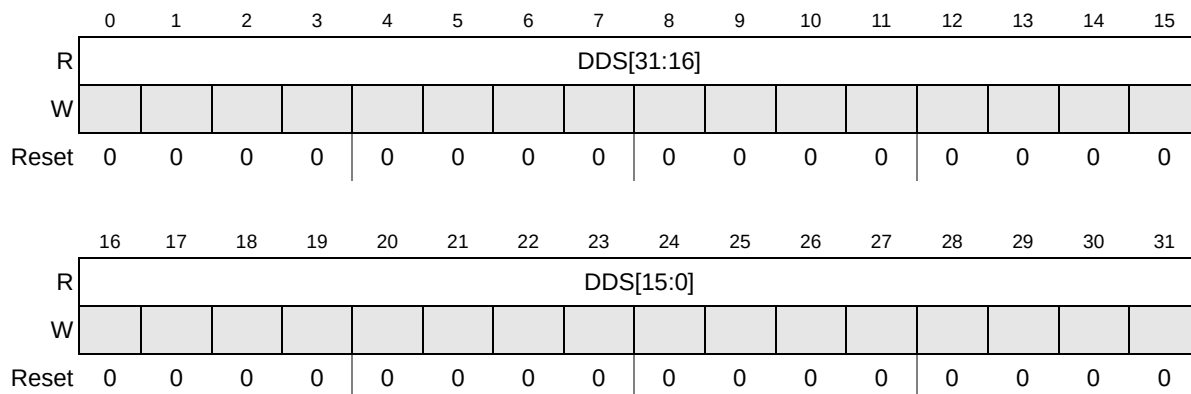


Table 1143. PSI5_DDSR field descriptions

Field	Description
0–31 DDS[31:0]	This register maps each of the individual Diagnostic registers (in the order NDSR, EISR) depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the Section 53.3.2.2, PSI5 Channel Control Register (PSI5_PCCR) , Section 53.3.2.3, DMA Control Register (PSI5_DCR) and Section 53.3.2.4, DMA Status Register (PSI5_DSR) .

Note: No FIFO is implemented for diagnostic registers. They are two 32-bit registers addressable by either the CPU (reading NDSR or EISR directly) or the DMA (reading the DDSR successively).

53.3.2.13 PSI5 Message Receive Register Low (PSI5_PMRRL)

[Figure 1139](#) shows the PSI5 Message Receive Register Low.

Figure 1139. PSI5 Message Receive Register Low (PSI5_PMRRL)

Address: Base + 0x0034 + (0x1C0 × channel)

Access: User read/write

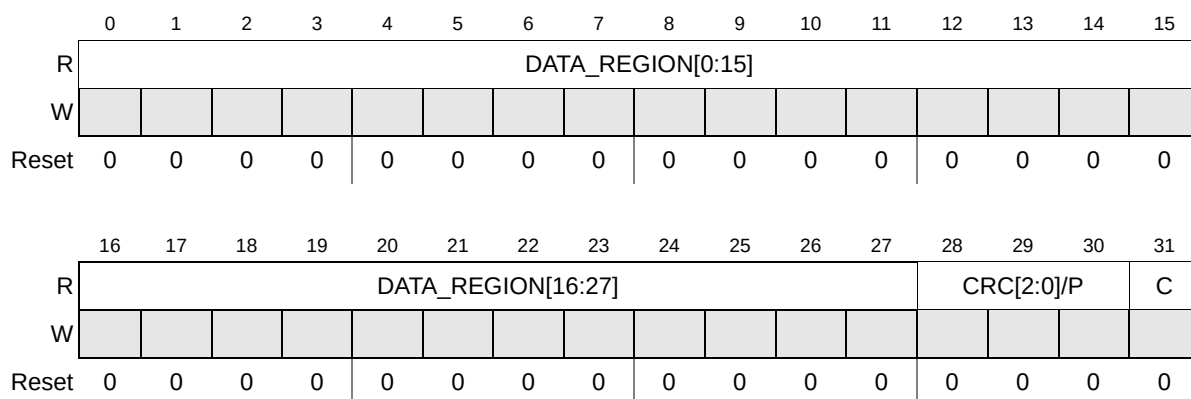


Table 1144. PSI5_PMRRL field description

Field	Description
0-27 DATA_REGION[0:27]	These are application-specific optional bits + data payload of received message. See Section 53.4.2.1, Reception of data frames for more details. The data is left-aligned, i.e., Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: $8 \leq n \leq 28$). If $n < 28$ remaining bit positions are filled with '0's, i.e., between last received bit and CRC field.
28–30 CRC[2:0]	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity.
31 C	This bit will be set if CRC/P recalculation return an Error.

53.3.2.14 PSI5 Message Receive Register High (PSI5_PMRRH)

Figure 1140 shows the PSI5 Message Receive Register High.

Figure 1140. PSI5 Message Receive Register High (PSI5_PMRRH)

Address: Base + 0x0038 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F	EM	E	T	Slot Counter[2:0]		Time Stamp Value[23:16]								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Time Stamp Value[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1145. PSI5_PMRRH field descriptions

Field	Description
0 Reserved	This bit is reserved.
1 F	This represents that NO frame was received in the corresponding configured slot. 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel (M0,M1), that is, at least one bit is absent during the T bit period. 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This bit indicates electrical error, i.e., at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). 0 No electrical error in fields other than M0, M1, and start fields. 1 Electrical error in fields other than M0, M1, and start fields.
4 T	This bit indicates timing error, i.e., frame has started in an unconfigured slot, spread across two slots, started before the first configured slot. 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.

Table 1145. PSI5_PMRRH field descriptions (continued)

Field	Description
5–7 Slot Counter[2:0]	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of “0”. For asynchronous modes the slot counter has a fixed value of “1”. The six slots are numbered as Slot1, Slot2,...Slot6.
8–31 Time Stamp Value[23:0]	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_SnFCR). In other words these bits contain either the start bit(S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where “F” bit has been set; the value of the Time Stamp appended to the message is “0”. It is possible to reset the time stamp counter by asserting to “1” the “gtm_reset” signal from the GTM.

Note: Any new PSI5 message is always stored in the PSI5 receive register (PMRRL/PMRRH). In parallel it is also stored in the RAM registers in a sequential manner like in a Ring Buffer. When a new message comes it would always overwrite the data in the PSI5 receive register. But the original message would still be available in the RAM register provided all locations in the RAM registers are not full, and an overwrite has not occurred.

53.3.2.15 PSI5 Message Register Low *i* (PSI5_PMRL*i*) {*i*: 0 to 31}

Figure 1141 shows the PSI5 Message Register Low *i*.

Figure 1141. PSI5 Message Register (Lower Word) {*i*:0 to 31} (PSI5_PMRL*n*)

Address: (0x0008+0x1C0*chnl)+(0x0034+i*0x8)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_REGION[0:15]															
W	IN THE NORMAL MODE, AUTO UPDATED BY HARDWARE AND NOT WRITEABLE BY ANY OTHER SOURCE. Writable in Config mode (for Test purposes).															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA_REGION[16:27]												CRC[2:0]/P			C
W	IN THE NORMAL MODE, AUTO UPDATED BY HARDWARE AND NOT WRITEABLE BY ANY OTHER SOURCE. Writable in Config mode (for Test purposes).															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1146. PSI5_PMRLn field descriptions

Field	Description
0–27 Data Region[0:27]	These are application-specific optional bits + data payload of received message. See Section 53.4.2.1, Reception of data frames for more details. The data is left-aligned, i.e., Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits ($n: 8 \leq n \leq 28$). If $n < 28$ remaining bit positions are filled with '0's, i.e., between last received bit and CRC field.
28-30 CRC[2:0]/P	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity.
31 C	This bit will be set if CRC/P recalculation return an Error.

53.3.2.16 PSI5 Message Register High *i* (PSI5_PMRHi) {i: 0 to 31}

[Figure 1142](#) shows the PSI5 Message Register High *i*.

Figure 1142. PSI5 Message Register High *n* (PSI5_PMRHn)

Address: $(0x0008 + 0x1C0 * chnl) + (0x0038 + i * 0x8)$

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	O	F	EM	E	T	Slot Counter[2:0]			Time Stamp Value[23:16]							
W		IN THE NORMAL MODE, AUTO UPDATED BY HARDWARE AND NOT WRITEABLE BY ANY OTHER SOURCE. Writable in Config mode (for Test purposes).														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Time Stamp Value[15:0]															
W	IN THE NORMAL MODE, AUTO UPDATED BY HARDWARE AND NOT WRITEABLE BY ANY OTHER SOURCE. Writable in Config mode (for Test purposes).															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1147. PSI5_PMRHn field descriptions

Field	Description
0 O	This bit carries the same information as OWSR, but appended here for each message separately. This bit is set when the current message (which is being read) has overwritten some previous unread message. Note that, if the corresponding location in the OWSR has been cleared or has been set by the SOWSR, then the same will be reflected in the message stored in the corresponding RAM location as well.
1 F	This represents that NO frame was received in corresponding configured Slot. 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).

Table 1147. PSI5_PMRHn field descriptions (continued)

Field	Description
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel (M0, M1), i.e., at least one bit is absent during the Tbit period. 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This Bit Indicates Electrical Error, i.e., at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). 0 No electrical error in fields other than M0, M1 and start fields. 1 Electrical error in fields other than M0, M1 and start fields.
4 T	This bit indicates timing error, i.e., frame has started in an unconfigured slot or it has spread across two slots or it has started before the first configured slot. 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 Slot Counter[2:0]	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of “0”. The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. For asynchronous modes the slot counter has a fixed value of “1”. The six slots are numbered as Slot1, Slot2,...Slot6.
8–31 Time Stamp Value[23:0]	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_SnFCR). In other words these bits contain either the start bit (S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where “F” bit has been set; the value of the Time Stamp appended to the message is “0”. It is possible to reset the time stamp counter by asserting to “1” the “gtm_reset” signal from the GTM.

53.3.2.17 SMC Frame Register n (PSI5_SFR n)

Figure 1143 shows the SMC Frame Register n .

Figure 1143. SMC Frame Register n (PSI5_SFR n)

Address: $(0x0008 + 0x1C0 * \text{chnl}) + (0x134 + (n-1) * 0x4)$

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLOT_NO			CER	OW	CRC[5:0] slow serial message CRC						C	ID[7:4]/ID[3:0]			
W	IN THE NORMAL MODE, AUTO UPDATED BY HARDWARE AND NOT WRITEABLE BY ANY OTHER SOURCE. Writable in Config mode (for Test purposes).					IN THE NORMAL MODE, AUTO UPDATED BY HARDWARE AND NOT WRITEABLE BY ANY OTHER SOURCE. Writable in Config mode (for Test purposes).										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ID[3:0]/DATA[15:12]				DATA[11:0]											
W	CANNOT BE WRITTEN IN THE NORMAL MODE.Writable in Config mode (for Test purposes).															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1148. PSI5_SFR n field descriptions

Field	Description
31–29 SLOT_NO	This indicates in which slot this SMC frame was received. Note that here the slot number refers to the slot in which all the re-arranged PSI5 messages (from which the M0,M1 bits have been extracted) are supposed to have occurred after the correct PSI5 message placement and Timing corrections have been performed. For SFR[1], slot_no = 1; For SFR[2], slot_no = 2; For SFR[3], slot_no = 3; For SFR[4], slot_no = 4; For SFR[5], slot_no = 5; For SFR[6], slot_no = 6;
3 CER	CRC Error 0 The present data has no CRC error. 1 The present data has a CRC error.

Table 1148. PSI5_SFRn field descriptions (continued)

Field	Description
4 OW	Overwrite status 0 The present data was already read when new message arrived. 1 New message has overwritten the previous unread message. <i>Note: If the corresponding SOWSM bits in the SSESr have been set or the OWSM bits in the GISR have been cleared, then the same will be reflected in the message stored in the corresponding RAM location as well. The overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten, the overwrite bit gets set.</i>
5-10 CRC	6-bit CRC for slow serial message
11 C	Configuration bit 0 12-bit data and 8-bit message ID. 1 16-bit data and 4-bit message ID.
12-15 ID[7:4]/ID[3:0]	Message ID: If C = '0' indicates ID[7:4], if C = '1' indicates ID[3:0].
16-19 ID[3:0]/DATA[15:12]	Message ID/DATA: If C = '0' indicates ID[3:0], if C = '1' indicates DATA[15:12].
20-31 DATA[11:0]	DATA payload

Note: Six dedicated 32-bit SFR[x] registers for SMC are available with a 1-1 slot correspondence. These can be read by either the CPU or the DMA (when DMA_EN_SF = 1).

- No FIFO for SMC frame is available. Six 32-bit registers are available in the address space, which can be accessed by either the DMA (using the DSFR) or the CPU (using the six SFR registers), depending on the setting of the DMA_EN_SMC bit.

53.3.2.18 New Data Status Register (PSI5_NDSR)

Figure 1144 shows the New Data Status Register.

Figure 1144. New Data Status Register (PSI5_NDSR)

Address: Base + 0x0154 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NDS[31:16]															
W	w1c or clear on read of corresponding PSI5 message															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NDS[15:0]															
W	w1c or clear on read of corresponding PSI5 message															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1149. PSI5_NDSR field descriptions

Field	Description
0–31 NDS[31:0]	New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when a new Message arrives, i.e., faulty or non faulty in corresponding PSI5 MB location These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SNDR.

53.3.2.19 Overwrite Status Register (PSI5_OWSR)

Figure 1145. Overwrite Status Register (OWSR)

Address: Base + 0x0158 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OWS[31:16]															
W	w1c or clear on read of corresponding PSI5 message															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OWS[15:0]															
W	w1c or clear on read of corresponding PSI5 message															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1150. OWSR field descriptions

Field	Description
0–31 OWS[31:0]	Over Write Status flags for PSI5 Messages corresponding to each PSI5 location in the RAM registers. These bits are set when any unread receive PSI5 message is overwritten by a newly arrived message). These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SOWSR.

53.3.2.20 Error Indication Status Register (PSI5_EISR)

Figure 1146 shows the Error Indication Status Register.

Figure 1146. Error Indication Status Register (PSI5_EISR)

Address: Base + 0x015C + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERROR[31:16]															
W	w1c or clear on read of corresponding PSI5 message															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERROR[15:0]															
W	w1c or clear on read of corresponding PSI5 message															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1151. PSI5_EISR field descriptions

Field	Description
0–31 ERROR[31:0]	Error Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when any of the five selectable (by ERROR_SELECT) bits (C, E, EM, T or F) is set for the corresponding PSI5 Message. These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independently by writing in SEISR.

53.3.2.21 Set New Data Status Register (PSI5_SNDSR)

Figure 1147 shows the Set New Data Status Register.

Figure 1147. Set New Data Status Register (PSI5_SNDSR)

Address: Base + 0x0160 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SNDS[31:16] (one shot writable to 1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SNDS[15:0] (one shot writable to 1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1152. PSI5_SNDSR field descriptions

Field	Description
0–31 SNDS[31:0]	Sets New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. The corresponding NDS Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. <i>Note: This bit is writable in config and Normal modes.</i>

53.3.2.22 Set Overwrite Status Register (PSI5_SOWSR)

Figure 1148 shows the Set Overwrite Status Register.

Figure 1148. Set Overwrite Status Register (PSI5_SOWSR)

Address: Base + 0x0164 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SOWS[31:16] (writable to 1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SOWS[15:0] (writable to 1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1153. PSI5_SOWSR field descriptions

Field	Description
0–31 SOWS[31:0]	Sets overwrite status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding OWS flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. <i>Note: These bits are writable in Config and Normal modes.</i>

53.3.2.23 Set Error Status Register (PSI5_SEISR)

Figure 1149 shows the Set Error Status Register.

Figure 1149. Set Error Status Register (PSI5_SEISR)

Address: Base + 0x0168 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SERROR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SERROR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1154. PSI5_SEISR field descriptions

Field	Description
0–31 SERROR[31:0]	Sets Error Status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding ERROR flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. <i>Note: These bits are writable in Config and Normal modes.</i>

53.3.2.24 Set SMC Error Status Register (PSI5_SSESR)

Figure 1150 shows the Set SMC Error Status Register.

Figure 1150. Set SMC Error Status Register (PSI5_SSESR)

Address: Base + 0x016C + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1155. PSI5_SSESR field description

Field	Description
2–7 SCESM[6:1]	Sets IS_CESM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_CESM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. <i>Note: These bits are writable in Config and Normal modes.</i>
18–23 SOWSM[6:1]	Sets IS_OWSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_OWSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. <i>Note: These bits are writable in Config and Normal modes.</i>
26–31 SNVSM[6:1]	Sets IS_NVSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_NVSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. <i>Note: This bit is writable in Config and Normal modes.</i>

53.3.2.25 Sync Time Stamp Read Register (PSI5_STSR)

Figure 1151 shows the Sync Time Stamp Read Register.

Figure 1151. Sync Time Stamp Read Register (PSI5_STSR)

Address: Base + 0x0170 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	STSV[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	STSV[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1156. PSI5_STSR field descriptions

Field	Description
8–31 STSV[23:0]	Sync Time Stamp Value[23:0] This bitfield provides the value of the free-running 24-bit time stamp counter triggered (captured) at last sync pulse.

53.3.2.26 Data Time Stamp Read Register (PSI5_DTSR)

Figure 1152 shows the Data Time Stamp Read Register.

Figure 1152. Data Time Stamp Read Register (DTSR)

Address: Base + 0x0174 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	SLOT_COUNTER[2:0]			DTSV[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTSV[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1157. DTSRR field descriptions

Field	Description
5–7 SLOT_COUNT ER[2:0]	3-bit Slot Counter, which corresponds to the slot for which the DTSRR contains the Time Stamp. The slot_counter contains the slot in which the start bits of this frame was captured (after successful detection of S1). If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of “0”. The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. In case of asynchronous mode, the slot counter has a fixed value of “1”.
8–15 DTSV[23:16]	Data Time Stamp Value[23:0] Whenever a valid sequence of the start bits (S0,S1) is detected, the value of the Time Stamp counter captured at the rising edge of the S0 is stored in this register at the rising edge of S1.

Note: If S0 and S1 are detected in different slots then the Time Stamp value and the Slot Counter, both will correspond to the first slot (S0)

53.3.2.27 Slot n Frame Configuration Register (PSI5_SnFCR)

Figure 1153 shows the Slot n Frame Configuration Register.

Figure 1153. Slot n Frame Configuration Register (PSI5_SnFCR)

Address: (0x0008+0x1C0*chnl)+0x0170 + (n-1)x0004

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SLOT_EN	TS_CAPT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SM	0	0	0	0	0	0	0	0	0	DRL[4:0]				CRC/P	
W	CL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1158. PSI5_SnFCR field descriptions

Field	Description
13 SLOT_EN	0 This particular slot is disabled, i.e., not in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) have no effect. 1 This particular slot is enabled, i.e., is in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) are applicable.
14 TS_CAPT	0 Time stamp value captured at rising edge of S0 and stored Start sequence first edge. 1 Time stamp value captured at corresponding Tsync pulse (posedge) and stored with corresponding slot PSI5 message.

Table 1158. PSI5_SnFCR field descriptions (continued)

Field	Description
16 SMCL	Serial Messaging Channel field 0 Serial Messaging Channel (optional) not present on Rx Message during Time Slot 'n' 12-bit{M1,M0} Serial Messaging Channel present on Rx Message during Time Slot 'n' This field is used to enable the detection of start sequence of SMC frame on particular slot. Detection is not done if this field is '0'.
26–30 DRL	Data Region Length[4:0] This field represents Length of Data region (including Frame Control, Status, Data payload1&2 regions and optional messaging bits M0,M1) in Rx Message during Time Slot 'n' 00000-00111 Invalid 01000 8 bit long data region 01001 9 bit long data region 01010 10 bit long data region 11011 127 bit long data region 11100 28 bit long data region 11101-11111 Invalid
31 CRC/P	0 3 bit CRC[2:0] present on Rx Message during Time Slot 'n' 1 1 bit Parity field present on Rx Message during Time Slot 'n'

Note: In Asynchronous mode, the PSI5_S1FCR register has to be programmed for controlling the various parameters pertaining to the message received. The other registers PSI5_S[2-6]FCR have no effect in the asynchronous mode.

53.3.2.28 Slot *n* Start Boundary Register (PSI5_SnSBR)

Figure 1154 shows the Slot *n* Start Boundary Register (PSI5_SnSBR).

Total of six registers for each of the six slots. These are S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR. Here *n* = 1 to 6 corresponding to each of the six slots.

Figure 1154. Slot *n* Start Boundary Register (PSI5_SnSBR)

Address: (0x0008+0x1C0*chnl)+0x0188 + (n-1)*0002

Access: User read/write

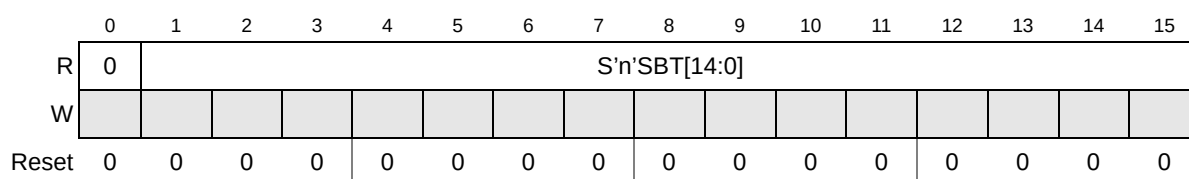


Table 1159. PSI5_SnSBR field descriptions

Field	Description
0–15 SnSBT[14:0]	Slot <i>n</i> Start Boundary Time This field specifies the time in intervals of 1 μs, occurring after the <i>rising edge</i> of the timing sync pulse, at which the slot <i>n</i> should start. <i>n</i> = 1 to 6 dedicated for each of the six slots.

53.3.2.29 Slot *n* End Boundary Register (PSI5_*Sn*EBR)

Figure 1154 shows the Slot *n* Start Boundary Register (PSI5_*Sn*SBR).

Slot *n* End Boundary register is a **single register**, which can be programmed to define the “end” of the *n*th slot. The *n* can be defined in the SLOT_NO field of this register.

There are seven boundary registers for the six slots. Six of these registers (S1SBR, S2SBR, S3SBR+, S4SBR, S5SBR, and S6SBR) are used to define the start of each of the six slots.

The seventh register *Sn*EBR (Slot *n* End Boundary register) defines the end of the *n*th slot. The *n* is programmed in the slot_no field of this register.

The number of slots would automatically be available from the *Sn*EBR. The *Sn*EBR contains the slot number of the last slot and the end boundary of the same. Each slot can separately be enabled or disabled by using the slot_en bit in the *Sn*FCR. When the start slot boundaries are properly defined in the *Sn*SBR (Slot *n* Start Boundary Register) register but the slot_en bit = 0 in the corresponding *Sn*FCR (Slot *n* Frame Configuration register) register, then the corresponding slot is treated as “available” but “unconfigured.” For such a slot the data/diagnostic bits all remain 0. *The “T” bit will, however, get set if a message that was supposed to come in a configured slot actually arrives in an unconfigured slot. Note that the various parameters like the Data length/CRC configuration and so on are valid ONLY for the configured slots.* Even when sampling a message in an unconfigured slot, these parameters would be taken from the previous/next configured slot only. It is assumed that no data would be sent in an unconfigured slot. If a data is received in this unconfigured slot it would never be treated as belonging to the unconfigured slot. As a special case if “S0” and “S1” are detected in different slots then a T-bit error is set.

Figure 1155. Slot *n* End Boundary Register (PSI5_*Sn*EBR)

Address: (0x0008+0x1C0*chnl)+0x0194

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SLOT_NO[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	<i>Sn</i> EBT[14:0]														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1160. PSI5_SnEBR field descriptions

Field	Description
13–15 SLOT_NO[2:0]	The field indicates the Slot number{1–6} for which Slot End Boundary Time is defined. Any value other than 1–6 is defaulted to 0. A value 0 here would mean that the Manchester decoder would not sample any slots. This slot number also defines the number of slots that the IP assumes to exist between 2 sync pulses.
17–31 SnEBT[14:0]	Slot <i>n</i> End Boundary Time. This field specifies the time in intervals of 1 μ s, occurring after the rising edge of timing sync pulse, at which the slot <i>n</i> should end.

Note: The start slot boundaries in the SnSBR should be correctly programmed. If this is not done, the correct operation cannot be guaranteed. Slots occurring after the slot_no value programmed in the SnEBR are treated as being unavailable (that is, the data in all the slots occurring after the SnEBR is rejected by the Manchester decoder).

Following are the examples:

- If only three slots are required between two sync pulses, program S1SBR, S2SBR, and S3SBR with the correct value of the start boundaries of the three slots and program slot_no = 3 in the SnEBR and define the end boundary of the third slot in SnEBR. The design ignores slots 4, 5, and 6.
- For six slots, the required programming would be: program the start of each of the six slots in S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR. Then program the end of the sixth slot in the SnEBR by putting slot_no = 6 and define the end boundary of the sixth slot in SnEBR.

53.3.2.30 Data Output Block Configuration Register (PSI5_DOBCCR)

Figure 1156 shows the Data Output Block Configuration Register.

Figure 1156. Data Output Block Configuration Register (PSI5_DOBCCR)

Address: Base + 0x01A0 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_LENGTH[5:0]					0	0	CMD_TYPE[2:0]				DEFAULT_SYNC	GTM_TRIG_SEL	SPULSE_SEL	OP_SEL	SW_READY
W						DBR_RST	DSR_RST									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1161. PSI5_DOBCCR field descriptions

Field	Description
0-5 DATA_LENGTH[5:0]	Can take on values from 0 to 63 corresponding to 1bit to 64bit non standard length commands. When PSI5_DOBCCR[CMD_TYPE] == 7(Non Standard), then this field controls the length of command that can be written to the DBR and the DSR registers, else the value in this field is ignored. This when DATA_LENGTH = 0 then the length of the command that can be written to DSR and the DBR is 1, when DATA_LENGTH=1 then the length of the command that can be written to DSR and the DBR is 2 and so on until a maximum of 64-bit length command when DATA_LENGTH = 63. Note that the length of the command that is programmed by using this field, is also equal to the length of the command that shifts out of the DSR register. In other words, it is equal to the number of bits, that shift out of DSR register once the DSR_RDY goes low. After these many shifts have happened, then the value corresponding to "DEFAULT_SYNC" starts getting shifted out and DSR_RDY goes as "1".
6 DBR_RST	This is to reset and reject current content to Data Buffer Register. When this bit is written as "1" then the contents of the DBR are reset to all "0s" (if DEFAULT_SYNC= 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DBR would be ready for new data. Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
7 DSR_RST	This is to reset and reject current content to Data Shift Register. When this bit is written as "1" then the contents of the DSR are reset to all "0s" (if DEFAULT_SYNC= 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DSR would be ready for new data. Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.

Table 1161. PSI5_DOBCCR field descriptions (continued)

Field	Description			
8-10 CMD_TYPE[2:0]	These 3 bits indicate the type of command that needs to be transmitted during the ECU to sensor communication. Following is a brief description of the same:			
	CMD_TYPE	PSI5 Frame	Effect on registers	Writeable Register lengths
	"000" - 0	Short Frame with 31 "1s" as the start condition*	Command can be written to DPR for auto calculation of stuff/start/CRC bits. Can always be overwritten in DSR/DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	6 bits in DPR 15 bits** in DBR/DSR.
	"001" - 1	Short Frame with 5 "0s" as the start condition*	-do-	-do-
	"010" - 2	Long Frame with 31 "1s" as the start condition*	-do-	16 bits in DPR. 29 bits** in DBR/DSR.
	"011" - 3	Long Frame with 5 "0s" as the start condition*	-do-	-do-
	"100" - 4	X-Long Frame with 31 "1s" as the start condition*	-do-	22 bits in DPR. 37 bits** in DBR/DSR
	"101" - 5	X-Long Frame with 5 "0s" as the start condition*.	-do-	-do-
	"110" - 6	XX-Long	-do-	24 bits in DPR. 43 bits** in DBR/DSR
	"111" - 7	Non Standard Length.	Command can only be written to DSR and DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	Programmable from 1 to 64 bits in DBR/DSR, depending on the value of PSI5_DOBCCR[DATA_LENGTH] field.
* The start condition (31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command; neither is this visible in any register. ** This is also the length of the command, that will be shifted out during the ECU to Sensor communication, irrespective of how many "actual" number of bits are written to DBR/DSR.				

Table 1161. PSI5_DOBCCR field descriptions (continued)

Field	Description
11 DEFAULT_SYNC	When this bit is set to “0” then the default value of DSR and DBR registers are all “0s”; when this bit is set to 1 then the default value of DBR and DSR registers are all “1s”. This value is used as the value of the default sync pulses shifted at the output path when the DSR_RDY == 1.
12 GTM_TRIG_SEL	GTM event triggered/internal sync pulse generator selection as shift clock for the DSR. 1GTM event shift triggered 0Internal sync pulse generator shift triggered
13 SP_PULSE_SEL	Selects the source for the short pulse PWM module: 1 SP module directly gets input from the GTM_event/internal sync pulse generator 0 SP module gets the data from the DSR
14 OP_SEL	This bit selects the driving source of the “ipp_do_psi5_sdout” port. 0 The sync pulse generator select as per the PSI5_DOBCCR[GTM_TRIG_SEL] 1 PWM output
15 SW_READY	When this bit is written to 1 the transfer from DBR to DSR automatically happens as soon as DSR_RDY becomes 1. When this bit is kept to 0 then the transfer from DBR to DSR will remain pending. Once the software makes this bit as “1” and DSR_RDY also goes as “1” the transfer to DSR takes place. However, note that when this bit is 0, and the DSR_RDY becomes 1, still the transfer to the DSR will NOT happen. Only when this bit is a 1, will the transfer happen. This bit can be written in CONFIG and the Normal modes.

Table 1162. DOBCCR bit configurations and the related output states

Output path states	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL
State1 ⁽¹⁾	0	1	1
State2 ⁽¹⁾	0	0	1
State3 ⁽¹⁾	1	1	1
State4 ⁽¹⁾	1	0	1
State5 ⁽¹⁾	1	x	0

1. These states are shown in [Section 53.8.2, Valid states for integrated sync pulse generator](#). In State5 the pulse width and period is to be controlled directly from the GTM. In this state, the PW0/1D registers have no effect on the Pulse Width Modulation, when observed on the “sdout” pin.

Note: Please refer to [Figure 1190](#) for the details about the above bits.

53.3.2.31 Manchester Decoder Disable Offset Register (PSI5_MDDIS_OFF)

The Manchester decoder Disable Offset register (PSI5_MDDIS_OFF) defines the time in intervals of 1 μ s, for which the Manchester decoder remains disabled after the falling edge of the sync pulse. Using this register, this time can be programmed from 0 μ s to 128 μ s. This works on the 1 μ s clock from the CGM.

Figure 1157. Manchester Decoder Disable Offset Register (PSI5_MDDIS_OFF)

Address: Base + 0x01A2 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	MDDIS_OFF						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1163. PSI5_MDDIS_OFF field descriptions

Field	Description
0–15 MDDIS_OFF	These 7 bits can be used to program the time for which the Manchester decoder remains disabled. This offset is added AFTER the falling edge of the sync pulse. Thus the total time for which the Manchester decoder remains disabled = Tsynch + MDDIS_OFF where Tsynch is the high time for the sync pulse. Thus Tsynch = Pulse_Width0 or Pulse_Width1 as the case maybe, depending on whether a “0” is being pulse modulated or a “1”. Note that the Manchester decoder will ALWAYS remain disabled during the high time of the Sync Pulse. Using this register it is possible to disable the Manchester decoder BEYOND the Sync Pulse High Time as well. Figure 1195 shows the “MDDIS_OFF” definition.

53.3.2.32 Pulse Width for Data Bit 0 Register (PSI5_PW0D)[Figure 1158](#) shows the Pulse Width for Data Bit 0 Register.**Figure 1158. PSI5_Pulse Width for Data Bit 0 Register (PW0D)**

Address: Base + 0x01A4 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	Pulse_Width0[6:0]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1164. PSI5_PW0D field descriptions

Field	Description
0–15 Pulse_Width0	This defines the width (in μs) of data value '0' to be send from Data Output Register. It is the number of clock cycles(1μs clock) counted up to width “Pulse_Width0”; starting as soon as trigger appears from the ISPG (Internal Sync Pulse Generator) or the GTM. Can take max value of 127. When, intending to use the” tooth gap method” of PSI5 command transmission, the Pulse_Width0 should be configured to 0. Figure 1195 shows the definition of “Pulse_Width0”.

53.3.2.33 Pulse Width for Data Bit 1 Register (PSI5_PW1D)

Figure 1159 shows the Pulse Width for Data Bit 1 Register.

Figure 1159. PSI5_Pulse Width for Data Bit 1 Register (PW1D)

Address	Base + 0x01A6 + (0x1C0 × channel)															Access: User read/write
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	Pulse_Width1[6:0]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1165. PSI5_PW1D field descriptions

Field	Description
0–15 Pulse_Width1	This defines the width (in μ s) of data value '1' to be send from Data Output Register. It is the number of clock cycles(1 μ s clock) counted up to width “Pulse_Width1”; starting as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM. Can take max value of 127. Figure 1195 shows the definition of “Pulse_Width1”.

Note: When using the “tooth gap method” of PSI5 command transmission, the pulse width is NOT disabled. In this case, PSI5_PW0D has to be programmed as 0. PSI5_PW1D has to have the value of the length of the pulse desired when a 1 has to be transferred.

53.3.2.34 Counter Target Pulse Register (PSI5_CTPR)

Figure 1160 shows the Counter Target Pulse Register.

Figure 1160. Counter Target Pulse Register (PSI5_CTPR)

Address:	Base + 0x01A8 + (0x1C0 × channel)															Access: User read/write
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CTPR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1166. PSI5_CTPR field descriptions

Field	Description
0–15 CTPR	Counter Target Pulse Register. This is the target counter value. Once the CTC (channel target counter) reaches the value of CTPR this CTC is reset. CTC is a 16-bit free running counter that starts/stops depending on the CTC_GED (in GCR) bit or the CTC_ED bit (in PCCR). It gets reset initially when its value reaches CIPR and subsequently when its value reaches CTPR. The fact that which out of CTC_GED or the CTC_ED bit enables/disables it, is dependent on the CTC_GED_SEL bit in the PCCR. When CTC_GED/CTC_ED == 0 then the CTC is reset. The minimum value of CTPR is 6. Values less than 6 will be defaulted to 0 and the CTPR counter will not start. The value in CTPR indicates the time period of the internally generated sync pulses. The unmodulated (No PWM) internally generated sync pulses have a high time of 4-sp_ts_clk cycles, i.e., the duty cycle of the unmodulated internally generated sync pulses is $(4/ CTPR * 100)$. The 4 sp_ts_clk cycles is just the nascent value of the pulse that, actually triggers the PWD counters Note that this gives one more possible state, in addition to the states shown in Section 53.8.2, Valid states for integrated sync pulse generator. This additional state allows the output of the internal sync pulse generator to be made directly available at the output, without being pulse modulated. The CTPR describes ONLY the SYNC pulse period and not the sync pulse length, which is governed by the PWD registers, except in State5 where in the GTM governs these parameters.

Note: Please see [Section 53.8.1, Internal SYNC Pulse generation and coordination across PSI5 channels](#) for details about the Sync Pulse generation and coordination.

53.3.2.35 Counter Initialize Pulse Register (PSI5_CIPR)

[Figure 1161](#) shows the Counter Initialize Pulse Register.

Figure 1161. Counter Initialize Pulse Register (PSI5_CIPR)

Address: Base + 0x01AA + (0x1C0 × channel)

Access: User read/write

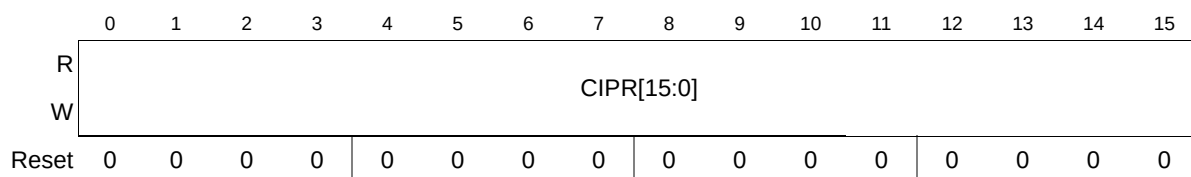


Table 1167. PSI5_CIPR field descriptions

Field	Description
0–15 CIPR[15:0]	Counter initialize pulse register. This is the value at which the CTC gets reset the first time, after which it is enabled using CTC_GED (in GCR) or the CTC_ED bit (in PCCR). All subsequent resets are done when the CTC value reaches CTPR. The CIPR is used to set the offset time between the start of sync pulses between two different PSI5 channels, when the CTC of both of these channels are started simultaneously using CTC_GED. The offset between any two PSI5 channels would be $CIPR_channel1 - CIPR_channel2$ sp_ts_clk cycles. Thus if channel 1 has a CIPR = 0 while channel2 has CIPR = 10, then the offset would be 10 sp_ts_clk cycles.

Note: The CIPR resolution is in terms of the time period of the “ sp_ts_clk ” ($T_{sp_ts_clk}$). Therefore the total “CIPR” time is “ $CIPR * T_{sp_ts_clk}$ ”, beginning from when the CIPR counters are enabled using the sync pulse trigger bits (PSI5_PCCR[CTC_ED] or PSI5_GCR[CTC_GED]). Due to internal Flop synchronization, this time would not exactly be $CIPR * T_{sp_ts_clk}$ but would have an inaccuracy of $1T_{sp_ts_clk}$. A further factor of $T_{ipg_clk_psi5}$ is also added to this inaccuracy.

The CTPR resolution is also in terms of the “ sp_ts_clk ”. Therefore the total “CTPR” time is “ $CTPR * T_{sp_ts_clk}$ ”. However when measured on the “sdout” pin of the IP this time would not exactly be “ $CTPR * T_{sp_ts_clk}$ ”. Due to internal Flop synchronization on the 1Mhz clock, an inaccuracy of “ $1\mu s$ ” would be added. A further factor of $T_{ipg_clk_psi5}$ is also added to this inaccuracy. Hence the “CTPR” period measured on the “sdout” pin, can range from $(CTPR * T_{sp_ts_clk}) - 1\mu s - T_{ipg_clk_psi5}$ to $(CTPR * T_{sp_ts_clk}) + 1\mu s + T_{ipg_clk_psi5}$. Please see [Figure 1195](#) for more details about this inaccuracy.

Please see [Section 53.8.1, Internal SYNC Pulse generation and coordination across PSI5 channels](#) for details about the Sync Pulse generation and coordination.

53.3.2.36 Data Preparation Register Low (PSI5_DPRL)

[Figure 1162](#) shows the Data Preparation Register Low.

Figure 1162. Data Preparation Register Low (PSI5_DPRL)

Address: Base + 0x01AC + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR
W									23	22	21	20	19	18	17	16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR	DPR
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1168. DPRL PSI5_field descriptions

Field	Description																		
8–31 DPR[23:0]	DPR[23:0] are the 24-bits of the DPR register used for writing the variable length, standard ECU-to-Sensor commands, comprising of the address, data and other fields. Note that the IP is transparent to the arrangement of the address, data and other fields and it treats the bits in these fields identically for the purpose of CRC calculation and transmission. This register is writable only when using the standard length PSI5 frames. For details about CRC calculation please refer to Section 53.4.3.4, Data transmission. The writable bits in this register are governed by the PSI5_DBOCR[CMD_TYPE] bit fields as per the following description: <table><tr><th>PSI5_DBOCR[CMD_TYPE]</th><th>Standard PSI5 FRAME Types</th><th>Writeable bits in PSI5_DPRL</th></tr><tr><td>“0” or “1”</td><td>Short Frame</td><td>6(DPR[5:0])</td></tr><tr><td>“2” or “3”</td><td>Long Frame</td><td>16(DPR[15:0])</td></tr><tr><td>“4” or “5”</td><td>X-Long Frame</td><td>22(DPR[21:0])</td></tr><tr><td>“6”</td><td>XX-Long Frame</td><td>24(DPR[23:0])</td></tr><tr><td>“7”</td><td>Not Available for write</td><td>Not Available for write</td></tr></table> Note that the unwriteable bits in this register field are always read as “0s”. This register has to be written by the CPU. These bits can be written only in the Normal mode.	PSI5_DBOCR[CMD_TYPE]	Standard PSI5 FRAME Types	Writeable bits in PSI5_DPRL	“0” or “1”	Short Frame	6(DPR[5:0])	“2” or “3”	Long Frame	16(DPR[15:0])	“4” or “5”	X-Long Frame	22(DPR[21:0])	“6”	XX-Long Frame	24(DPR[23:0])	“7”	Not Available for write	Not Available for write
PSI5_DBOCR[CMD_TYPE]	Standard PSI5 FRAME Types	Writeable bits in PSI5_DPRL																	
“0” or “1”	Short Frame	6(DPR[5:0])																	
“2” or “3”	Long Frame	16(DPR[15:0])																	
“4” or “5”	X-Long Frame	22(DPR[21:0])																	
“6”	XX-Long Frame	24(DPR[23:0])																	
“7”	Not Available for write	Not Available for write																	

Note: This register is the lower register used for writing the standard data and the address commands and is to be used for writing standard length ECU-to-sensor commands. The CRC/stuff/start bits are automatically appended by the hardware before the command is transferred to DBR. The commands to this register are accepted ONLY when $PSI5_GISR[DPR_RDY] == 1$. Whenever $PSI5_GISR[DPR_RDY] == 1$, then each bit of this register defaults to 0. Writing to this register when $PSI5_GISR[DPR_RDY] == 0$ will cause the $PSI5_GISR[IS_PROW]$ status bit to be set and the generation of an error interrupt, if it is enabled. Further, trying to write to this register when $PSI5_DOBCR[CMD_TYPE] = “7”$ results in the setting of the $PSI5_GISR[IS_PROW]$ bit and the resulting write is rejected.

53.3.2.37 Data Preparation Register High (PSI5_DPRH)

Figure 1163. Data Preparation Register High (PSI5_DPRH)

Address: Base + 0x01B0 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1169. PSI5_DPRH field descriptions

Field	Description
0–31 DPRH[31:0]	This field defines the upper word of 64-bit Preparation Register. This register is always read as 0s. Writing to this register does not have any effect.

53.3.2.38 Data Buffer Register Low (PSI5_DBRL)

Figure 1164 shows the Data Buffer Register Low.

Figure 1164. Data Buffer Register Low (PSI5_DBRL)

Address: Base + 0x01B4 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBR [31]	DBR [30]	DBR [29]	DBR [28]	DBR [27]	DBR [26]	DBR [25]	DBR [24]	DBR [23]	DBR [22]	DBR [21]	DBR [20]	DBR [19]	DBR [18]	DBR [17]	DBR [16]
W	CAN BE WRITTEN BY THE CPU IN THE NORMAL WHEN PSI5_GISR[DBR_RDY] == 1. CANNOT BE WRITTEN IN THE CONFIG MODE.															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DBR [15]	DBR [14]	DBR [13]	DBR [12]	DBR [11]	DBR [10]	DBR [9]	DBR [8]	DBR [7]	DBR [6]	DBR [5]	DBR [4]	DBR [3]	DBR [2]	DBR [1]	DBR [0]
W	CAN BE WRITTEN BY THE CPU IN THE Normal MODE WHEN PSI5_GISR[DBR_RDY] == 1. CANNOT BE WRITTEN IN THE CONFIG MODE.															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1170. PSI5_DBRL field descriptions

Field	Description																			
0-31 DBR[31:0]	This register contains the Lower 32 bits(DBR[31:0]) of the max-64 bit length command (DBR[63:0]). The higher bits, DBR[63:32] are contained in the DBRH register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU (when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware update occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in the below tables The tables (except when PSI5_DOBCR[CMD_TYPE] = “7”) show the bit assignment of the DBR[31:0] when these bits are automatically updated by the hardware. When PSI5_DOBCR[CMD_TYPE] == “7”), then only the CPU can write to these registers.																			
	<table><tr><th>PSI5_DOBCR [CMD_TYPE]</th><th>DBR bits</th><th>Bit Assignment</th></tr><tr><td>“7” (Non Standard Length)</td><td>DBR[31:0]]</td><td>Variable length of writeable bits, as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]</td></tr></table>	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment	“7” (Non Standard Length)	DBR[31:0]]	Variable length of writeable bits, as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]													
	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment																	
	“7” (Non Standard Length)	DBR[31:0]]	Variable length of writeable bits, as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]																	
<table><tr><th>PSI5_DOBCR [CMD_TYPE]</th><th>DBR bits</th><th>Bit Assignment</th></tr><tr><td rowspan="8">“6”(XX Long)</td><td>DBR[31:30]</td><td>DPR[19:18]</td></tr><tr><td>DBR[29]</td><td>1'b0(stuff)</td></tr><tr><td>DBR[28:23]</td><td>DPR[17:12]</td></tr><tr><td>DBR[22]</td><td>1'b0(stuff)</td></tr><tr><td>DBR[21:16]</td><td>DPR[11:6]</td></tr><tr><td>DBR[15]</td><td>1'b0(stuff)</td></tr><tr><td>DBR[14:9]</td><td>DPR[5:0]</td></tr><tr><td>DBR[8:0]</td><td>"011111110"</td></tr></table>	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment	“6”(XX Long)	DBR[31:30]	DPR[19:18]	DBR[29]	1'b0(stuff)	DBR[28:23]	DPR[17:12]	DBR[22]	1'b0(stuff)	DBR[21:16]	DPR[11:6]	DBR[15]	1'b0(stuff)	DBR[14:9]	DPR[5:0]	DBR[8:0]	"011111110"
PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment																		
“6”(XX Long)	DBR[31:30]	DPR[19:18]																		
	DBR[29]	1'b0(stuff)																		
	DBR[28:23]	DPR[17:12]																		
	DBR[22]	1'b0(stuff)																		
	DBR[21:16]	DPR[11:6]																		
	DBR[15]	1'b0(stuff)																		
	DBR[14:9]	DPR[5:0]																		
	DBR[8:0]	"011111110"																		

Table 1170. PSI5_DBRL field descriptions (continued)

Field	Description		
0-31 DBR[31:0] (Continued)	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
	“4” or “5” (X-Long)	DBR[31]	1'b1(stuff)
		DBR[30:28]	DPR[20:18]
		DBR[27]	1'b1(stuff)
		DBR[26:24]	DPR[17:15]
		DBR[23]	1'b1(stuff)
		DBR[22:20]	DPR[14:12]
		DBR[19]	1'b1(stuff)
		DBR[18:16]	DPR[11:9]
		DBR[15]	1'b1(stuff)
		DBR[14:12]	DPR[8:6]
		DBR[11]	1'b1(stuff)
		DBR[10:8]	DPR[5:3]
		DBR[7]	1'b1(stuff)
		DBR[6:4]	DPR[2:0]
		DBR[3]	1'b1(stuff)
		DBR[2:0]	“010”(start seq)
	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
	“2” or “3” (Long)	DBR[31:29]	“000”
		DBR[28]	CRC[0]
		DBR[27]	1'b1(stuff)
		DBR[26:24]	{CRC[1],CRC[2],D PR[15]}
		DBR[23]	1'b1(stuff)
		DBR[22:20]	DPR[14:12]
		DBR[19]	1'b1(stuff)
		DBR[18:16]	DPR[11:9]
		DBR[15]	1'b1(stuff)
		DBR[14:12]	DPR[8:6]
		DBR[11]	1'b1(stuff)
		DBR[10:8]	DPR[5:3]
		DBR[7]	1'b1(stuff)
		DBR[6:4]	DPR[2:0]
		DBR[3]	1'b1(stuff)
		DBR[2:0]	“010”(start seq)

Table 1170. PSI5_DBRL field descriptions (continued)

Field	Description		
0-31 DBR[31:0] (Continued)	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
	“0” or “1” (Short)	DBR[31:15]	0
		DBR[14:12]	{CRC[0],CRC[1],C RC[2]}[
		DBR[11]	1'b1(stuff)
		DBR[10:8]	DPR[5:3]
		DBR[7]	1'b1(stuff)
		DBR[6:4]	DPR[2:0]
		DBR[3]	1'b1(stuff)
		DBR[2:0]	“010”(start seq)

This register is writable only in Normal mode.

Note: When `DEFAULT_SYNC == 0` then default value of this register is `00000000`. When `DEFAULT_SYNC == 1` then the default value of this register is `FFFFFFF`. Note that the default value is different from the reset value. The reset value of this register is always “00000000”. The default value is loaded once the IP enters the NORMAL mode.

Further, the commands to this register are accepted ONLY when `PSI5_GISR[DBR_RDY] == 1`. Whenever `PSI5_GISR[DBR_RDY] == 1` then each bit of this register defaults to `DEFAULT_SYNC` value. Writing to this register when `PSI5_GISR[DBR_RDY] == 0` will set the status bit `PSI5_GISR[IS_BROW]` and generate an error interrupt, if it is enabled.

53.3.2.39 Data Buffer Register High (PSI5_DBRH)

Figure 1165 shows the Data Buffer Register High.

Figure 1165. Data Buffer Register High (PSI5_DBRH)

Address	Base + 0x01B8 + (0x1C0 × channel)																Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15																
R	DBR [63]	DBR [62]	DBR [61]	DBR [60]	DBR [59]	DBR [58]	DBR [57]	DBR [56]	DBR [55]	DBR [54]	DBR [53]	DBR [52]	DBR [51]	DBR [50]	DBR [49]	DBR [48]																
W	CAN BE WRITTEN BY THE CPU IN THE Normal MODE WHEN PSI5_GISR[DBR_RDY] == 1. CANNOT BE WRITTEN IN THE CONFIG MODE.																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
R	DBR [47]	DBR [46]	DBR [45]	DBR [44]	DBR [43]	DBR [42]	DBR [41]	DBR [40]	DBR [39]	DBR [38]	DBR [37]	DBR [36]	DBR [35]	DBR [34]	DBR [33]	DBR [32]																
W	CAN BE WRITTEN BY THE CPU IN THE Normal MODE WHEN PSI5_GISR[DBR_RDY] == 1. CANNOT BE WRITTEN IN THE CONFIG MODE.																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

Table 1171. PSI5_DBRH field descriptions

Field	Description																																				
0—31 DBR[63:32]	This register contains the Upper 32 bits(DBR[63:32]) of the max-64 bit length command (DBR[63:0]). The lower bits, DBR[31:0] are contained in the DBRL register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU (when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware update occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in the below tables. Note that when PSI5_DOBCR[CMD_TYPE] == “7”), then only the CPU can write to these registers. <table><tr><th>PSI5_DOBCR [CMD_TYPE]</th><th>DBR bits</th><th>Bit Assignment</th></tr><tr><td>“7” - Non Standard Frame lengths</td><td>DBR[63:32]]</td><td>Variable length of writeable bits, as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]</td></tr></table> <table><tr><th>PSI5_DOBCR [CMD_TYPE]</th><th>DBR bits</th><th>Bit Assignment</th></tr><tr><td rowspan="4">“6” - XX Long</td><td>DBR[63:43]</td><td>All 0s</td></tr><tr><td>DBR[42:37]</td><td>{CRC[5],CRC[4],CRC[3],CRC[2],CRC[1],CRC[0]}</td></tr><tr><td>DBR[36]</td><td>1'b0(stuff)</td></tr><tr><td>DBR[32:35]</td><td>DPR[23:20]</td></tr></table> <table><tr><th>PSI5_DOBCR [CMD_TYPE]</th><th>DBR bits</th><th>Bit Assignment</th></tr><tr><td rowspan="4">“4” or “5” -X Long</td><td>DBR[63:37]</td><td>All 0s</td></tr><tr><td>DBR[36]</td><td>CRC[0]</td></tr><tr><td>DBR[35]</td><td>1'b1((stuff)</td></tr><tr><td>DBR[34:32]</td><td>{CRC[1],CRC[2],DPR[21]}</td></tr></table> <table><tr><th>PSI5_DOBCR [CMD_TYPE]</th><th>DBR bits</th><th>Bit Assignment</th></tr><tr><td>“3” or “2”(Long) OR “1” or “0”(Short)</td><td>DBR[63:32]</td><td>All 0s</td></tr></table>	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment	“7” - Non Standard Frame lengths	DBR[63:32]]	Variable length of writeable bits, as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment	“6” - XX Long	DBR[63:43]	All 0s	DBR[42:37]	{CRC[5],CRC[4],CRC[3],CRC[2],CRC[1],CRC[0]}	DBR[36]	1'b0(stuff)	DBR[32:35]	DPR[23:20]	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment	“4” or “5” -X Long	DBR[63:37]	All 0s	DBR[36]	CRC[0]	DBR[35]	1'b1((stuff)	DBR[34:32]	{CRC[1],CRC[2],DPR[21]}	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment	“3” or “2”(Long) OR “1” or “0”(Short)	DBR[63:32]	All 0s
	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment																																		
	“7” - Non Standard Frame lengths	DBR[63:32]]	Variable length of writeable bits, as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]																																		
	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment																																		
	“6” - XX Long	DBR[63:43]	All 0s																																		
		DBR[42:37]	{CRC[5],CRC[4],CRC[3],CRC[2],CRC[1],CRC[0]}																																		
		DBR[36]	1'b0(stuff)																																		
		DBR[32:35]	DPR[23:20]																																		
	PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment																																		
	“4” or “5” -X Long	DBR[63:37]	All 0s																																		
DBR[36]		CRC[0]																																			
DBR[35]		1'b1((stuff)																																			
DBR[34:32]		{CRC[1],CRC[2],DPR[21]}																																			
PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment																																			
“3” or “2”(Long) OR “1” or “0”(Short)	DBR[63:32]	All 0s																																			

This register is CPU writable only in Normal mode.

Note: When `DEFAULT_SYNC == 0` then the default value of this register is `00000000`. When `DEFAULT_SYNC == 1` then the default value of this register is `FFFFFFF`. Note that the default value is different from the reset value. The reset value of this register is always `"00000000"`. The default value is loaded once the IP enters the NORMAL mode.

Further, the commands to this register are accepted ONLY when `PSI5_GISR[DBR_RDY] == 1`. Whenever `PSI5_GISR[DBR_RDY] == 1` then each bit of this register defaults to `DEFAULT_SYNC` value. Writing to this register when `PSI5_GISR[DBR_RDY] == 0` will set the status bit `PSI5_GISR[IS_BROW]` and generate an error interrupt, if it is enabled.

53.3.2.40 Data Shift Register Low (PSI5_DSRL)

Figure 1166 shows the Data Shift Register Low.

Figure 1166. Data Shift Register Low (PSI5_DSRL)

Address: Base + 0x01BC + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSR [31]	DSR [30]	DSR [29]	DSR [28]	DSR [27]	DSR [26]	DSR [25]	DSR [24]	DSR [23]	DSR [22]	DSR [21]	DSR [20]	DSR [19]	DSR [18]	DSR [17]	DSR [16]
W	CAN BE WRITTEN BY THE CPU IN THE Normal MODE WHEN <code>PSI5_GISR[DBR_RDY] == 1</code> . CANNOT BE WRITTEN IN THE CONFIG MODE.															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DSR [15]	DSR [14]	DSR [13]	DSR [12]	DSR [11]	DSR [10]	DSR [9]	DSR [8]	DSR [7]	DSR [6]	DSR [5]	DSR [4]	DSR [3]	DSR [2]	DSR [1]	DSR [0]
W	CAN BE WRITTEN BY THE CPU IN THE Normal MODE WHEN <code>PSI5_GISR[DBR_RDY] == 1</code> . CANNOT BE WRITTEN IN THE CONFIG MODE.															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1172. PSI5_DSRL field descriptions

Field	Description
0-31 DSR[31:0]	These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of <code>PSI5_DOBDR[CMD_TYPE[2:0]]</code> register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit positions in this register and the bit positions in the DBRL. Thus when updated by the hardware, then <code>DSR[31:0] = DBR[31:0]</code>. Hardware update occurs when <code>PSI5_GISR[DSR_RDY] == 1</code> and <code>PSI5_GISR[DBR_RDY] == 0</code> and <code>PSI5_DOBDR[SW_READY] == 1</code>. Note that if <code>PSI5_DOBDR[SW_READY] == 0</code> then the update of the data from the DBR to DSR will NOT happen even though <code>PSI5_GISR[DSR_RDY] == 1</code>. These bits can be written by the CPU in the Normal mode, when <code>PSI5_GISR[DSR_RDY] == 1</code>.

This register is CPU writable only in Normal mode.

Note: When `DEFAULT_SYNC == 0` then default value of this register is `00000000`. When `DEFAULT_SYNC == 1` then the default value of this register is `FFFFFFF`. Note that the default value is different from the reset value. The reset value of this register is always “00000000”. The default value is loaded once the IP enters the NORMAL mode.

Further, the commands to this register are accepted ONLY when `PSI5_GISR[DSR_RDY] == 1`. Whenever `PSI5_GISR[DBR_RDY] == 1` then each bit of this register defaults to `DEFAULT_SYNC` value. Writing to this register when `PSI5_GISR[DSR_RDY] == 0` will set the status bit `GISR[IS_DSROW]` and generate an error interrupt, if it is enabled. Note that for the hardware update of DSR with DBR contents, additional condition `PSI5_DOBCR[SW_READY] == 1`, should also be satisfied in addition to `PSI5_GISR[DSR_RDY] == 1`.

53.3.2.41 Data Shift Register High (PSI5_DSRH)

Figure 1167 shows the Data Shift Register High.

Figure 1167. Data Shift Register Low (PSI5_DSRH)

Address: Base + 0x01CS0 + (0x1C0 × channel)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSR [63]	DSR [62]	DSR [61]	DSR [60]	DSR [59]	DSR [58]	DSR [57]	DSR [55]	DSR [55]	DSR [54]	DSR [53]	DSR [52]	DSR [51]	DSR [50]	DSR [49]	DSR [48]
W	CAN BE WRITTEN BY THE CPU IN THE Normal MODE WHEN <code>PSI5_GISR[DSR_RDY] == 1</code> . CANNOT BE WRITTEN IN THE CONFIG MODE.															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DSR [47]	DSR [46]	DSR [45]	DSR [44]	DSR [43]	DSR [42]	DSR [41]	DSR [40]	DSR [39]	DSR [38]	DSR [37]	DSR [36]	DSR [35]	DSR [34]	DSR [33]	DSR [32]
W	CAN BE WRITTEN BY THE CPU IN THE NORMAL MODE WHEN <code>PSI5_GISR[DSR_RDY] == 1</code> . CANNOT BE WRITTEN IN THE CONFIG MODE.															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1173. PSI5_DSRH field descriptions

Field	Description
0-31 DSR[63:32]	These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of <code>PSI5_DOBCR[CMD_TYPE[2:0]]</code> register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit positions in this register and the bit positions in the DBRH. Thus when updated by the hardware, then <code>DSR[63:32] = DBR[63:32]</code>. Hardware update occurs when <code>PSI5_GISR[DSR_RDY] == 1</code> and <code>PSI5_GISR[DBR_RDY] == 0</code> and <code>PSI5_DOBCR[SW_READY] == 1</code>. Note that if <code>PSI5_DOBCR[SW_READY] == 0</code> then the update of the data from the DBR to DSR will NOT happen even though <code>PSI5_GISR[DSR_RDY] == 1</code>. These bits can be written by the CPU in the Normal mode, when <code>PSI5_GISR[DSR_RDY] == 1'b1</code>.

This register is writable only in Normal mode.

Note: When `DEFAULT_SYNC == 0` then the default value of this register is `00000000`. When `DEFAULT_SYNC == 1` then the default value of this register is `FFFFFFFF`. Note that the default value is different from the reset value. The reset value of this register is always “00000000”. The default value is loaded once the IP enters the NORMAL mode.

Further, the commands to this register are accepted ONLY when `PSI5_GISR[DSR_RDY] == 1`. Whenever `PSI5_GISR[DBR_RDY] == 1` then each bit of this register defaults to `DEFAULT_SYNC` value. Writing to this register when `PSI5_GISR[DSR_RDY] == 0` will set the status bit `GISR[IS_DSROW]` and generate an error interrupt, if it is enabled. Note that for the hardware update of DSR with DBR contents, additional condition `PSI5_DOBCR[SW_READY] == 1`, should also be satisfied in addition to `PSI5_GISR[DSR_RDY] == 1`.

53.3.2.42 Device modes and Register bit accesses

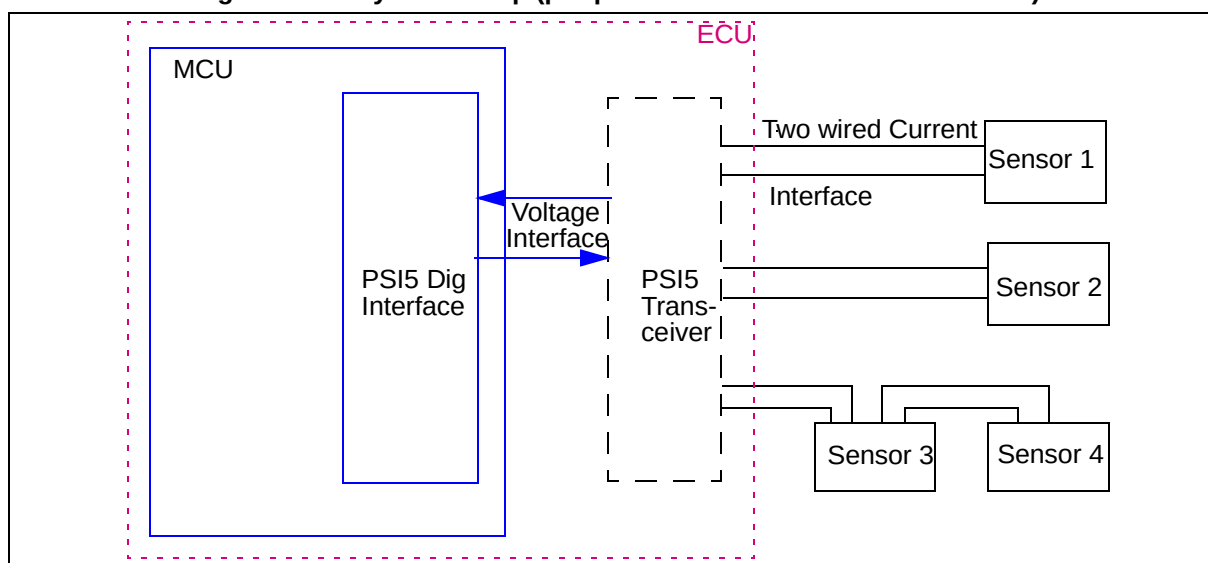
1. The following bits are writable in all modes (Normal, Config, and Disable).
 - `GCR[GLOBAL_DISABLE_REQ]`
 - `PCCR[PSI5_CH_CONFIG]`
 - `PCCR[PSI5_CH_EN]`
 - Global Counter Enable/Disable (`GCR[CTC_GED]`). This is to allow the Sync Pulse Generator to be enabled/disabled independently among those channels that are used for staggering, irrespective of the state of the channels NOT being used for staggering.
2. The following register is writable only in Normal mode and is initialized to all 0s.
 - `DPRL`
3. The following registers are writable ONLY in Normal mode. Initial is at ion in Config mode depends on the value of `PSI5_DOBCR[DEFAULT_SYNC]` bit. When `PSI5_DOBCR[DEFAULT_SYNC] == 1` in Config mode, then the below registers are initialized to all 1s in Normal mode. When `PSI5_DOBCR[DEFAULT_SYNC] = 0` in Config mode, then the below registers are initialized to all 0s in Normal mode. Once in Normal mode, these registers can then be programmed for any command that needs to be sent.
 - `DBRH/DBRL`
 - `DSRH/DSRL`
4. The following registers are writable in Config and Normal modes only:
 - All interrupt control registers
 - All SET registers (the one shot registers that are used to set the error flags for software drivers)
 - Channel Enable/Disable (`PCCR[CTC_ED]`). Using this it is possible to allow the Sync Pulse Generator to be enabled/disabled in Normal mode. The counter is reset when `CTC_ED == 0`
 - All w1c bits
5. Apart from these, all other register bits can be written only in Config mode, including the Data registers (`PMRH/PMRL/SFRn`), that, in Normal mode, are updated by the hardware only.
6. Debug mode: If the device is in Normal mode and the debug mode is enabled, then ONLY the registers in [Section 53.3.2.42, Device modes and Register bit accesses](#):^{(1),(2),(3),(4)} can be written by the debugger. ⁽⁵⁾ of [Section 53.3.2.42, Device modes and Register bit accesses](#) will not be available for write through the debugger.

In case the debugger has to change any register, then the device has to be brought in the Disable → Config mode. The debugger can, however, read any register in any mode.

53.4 Functional description

The PSI5 is a communication protocol for peripheral inertial, pressure, temperature, and position sensors. These sensors are connected to the ECU by just two wires, using the same lines for power supply and data transmission. The transceiver (present externally) provides a pre-regulated voltage to the sensors and reads in the transmitted sensor data. [Figure 1168](#) shows a point-to-point connection for sensors 1 and 2 and bus configuration for sensors 3 and 4.

Figure 1168. System setup (peripheral sensor connected to device)



ECU (Electronic Control Unit) is comprised of PSI5 digital interface and the PSI5 transceiver (present externally).

53.4.1 Sensor-to-ECU communication

53.4.1.1 Physical layer

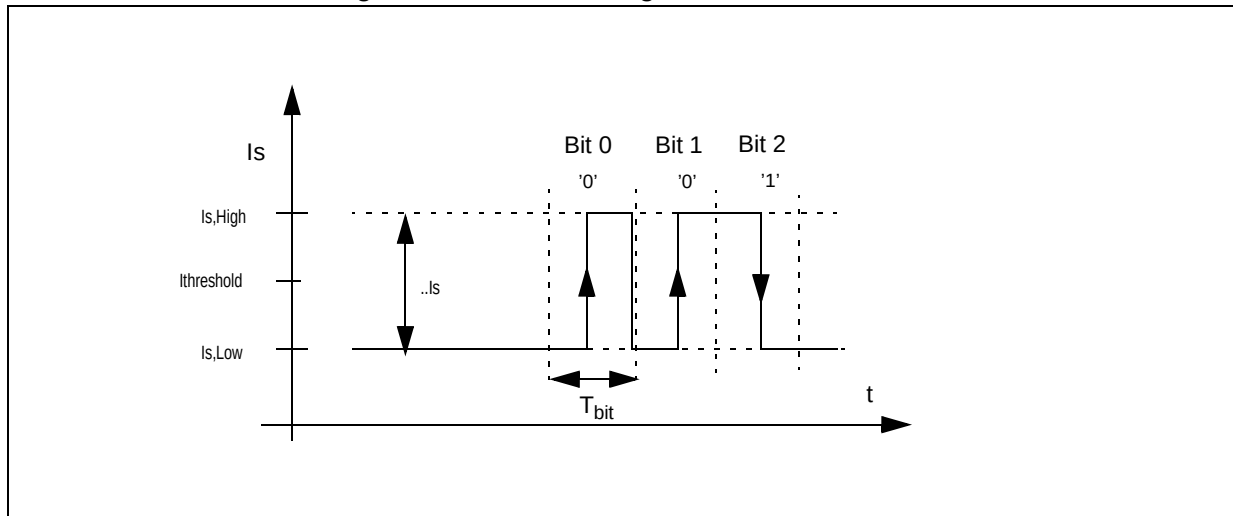
The PSI5 uses two wires to provide both power supply and data transmission to the sensors. The ECU provides a pre-regulated voltage to the sensors. Data transmission from the sensor to the ECU is done by current modulation on the power supply lines. Current oscillations are damped by the ECU.

53.4.1.2 Bit encoding

A “low” level ($I_{S,Low}$) is represented by the Normal (quiescent) current consumption of the sensor(s). A “high” level ($I_{S,High}$) is generated by an increased current sink of the sensor ($I_{S,Low} + \dots I_S$). The current modulation is detected within the PSI5 transceiver.

Manchester coding is used for data transmission. A logic 0 is represented by a rising slope and a logic 1 by a falling slope of the current in the middle of T bit.

Figure 1169. Bit encoding of data from sensor



53.4.1.3 Synchronization

For synchronized operation and bus mode, a sync pulse is generated by the PSI5 receiver. The sync pulse is output from the integrated PSI5 receiver to an external PSI5 transceiver that processes these pulses to a pre-regulated voltage level on supply lines to sensors. The voltage change is detected within the sensors.

53.4.2 Data Link Layer

Data bit decoding

The PSI5 message is received as serial Manchester-coded bits. The bit stream is passed through a Manchester decoder stage that converts it to normal NRZ format bit by bit and stores it serially in the receive register (64 bits). This block clearly identifies start (00) and end of receive frame using preprogrammed frame configuration registers.

- The Manchester-coded bits can be received at the rate of 125 Kbit/s or 189 Kbit/s (selectable by programming frame configuration registers).
- The Manchester decoder is designed to consider data with duty cycle of 30–70%.
- The bit decoding is done using an oversampling and counting algorithm with majority voting and edge recognition. $(125/189 \text{ kHz}) \times 32$, i.e., (4/6.048 MHz also referred as 32× clock) clocks are supplied from a common clock generator (to be implemented in CGM) module. This clock is used for oversampling of the receive bit and thereby decoding it as normal 1 or 0 (NRZ).
- The Manchester decoder is masked (switched off) for the duration when the sync pulse is transmitted. In addition to this, it can be configured to be off for an additional time, after the falling edge of the sync pulse. The duration of this time is configurable in the MDDIS_OFF register. It uses a 2003 voter to recognize the encoded half bits.
- There is a 3005 majority voting glitch filter at the data input of the Manchester decoder.

53.4.2.1 Reception of data frames

The extended PSI5 (Power Train Applications) receive data frame consists of p bits containing:

- Two start bits (S1 and S2) {S1=0, S2=0}
- One parity bit (P) with even parity (or 3 CRC bits C0, C1, and C2).
- A data region (D0.....D[$k - 1$]) with $k = 8$ to 28 bits with 1-bit granularity.
- The total length of a PSI5 frame is $p = k + 3$ data bits (in case of frames with parity bit) or $p = k + 5$ data bits (in case of frames with CRC).
- Data bits are received LSB first. The parity or CRC check bits cover the bits of the entire data region.

The length of the data region can vary between $k = 8$ to 28 bits (with 1-bit granularity). The data region can be split into the following fields and regions:

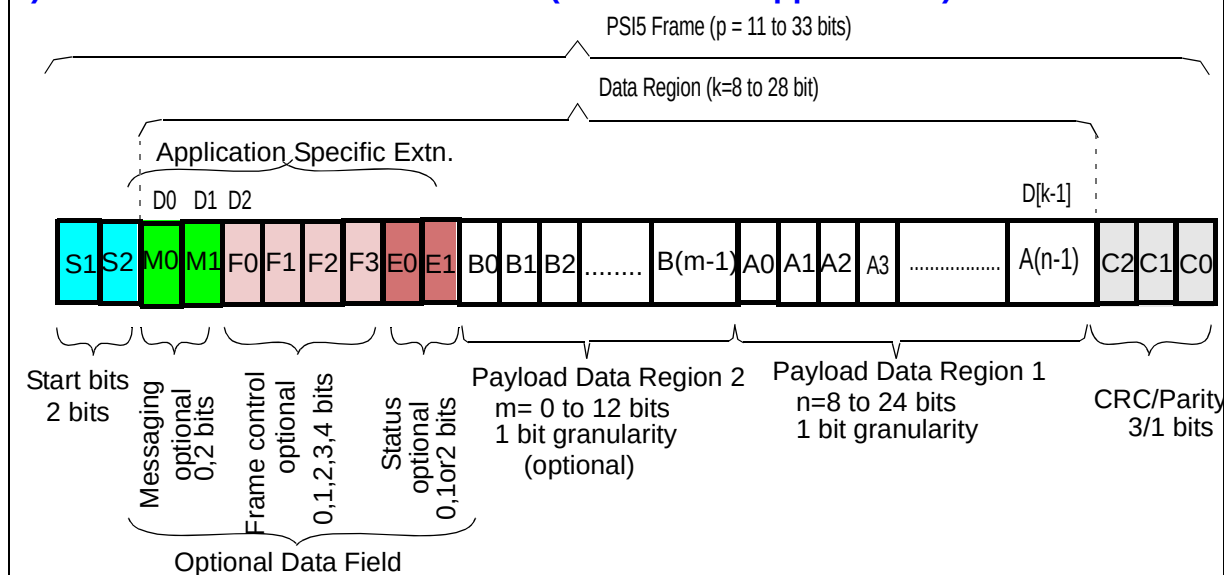
- The bits D0 and D1 can optionally represent Serial Messaging Channel M0, M1.
 - This Serial Messaging Channel (SMC) is used to receive initialization data during fast startup phase or slow acknowledgement/response messages (to command transmitted from ECU requesting data or diagnostic information from the sensor).

Note, that the maximum length of the entire data region is 28 bits. Thus the signal payload region and the optional data fields cannot concurrently be used at their maximum length in some cases.

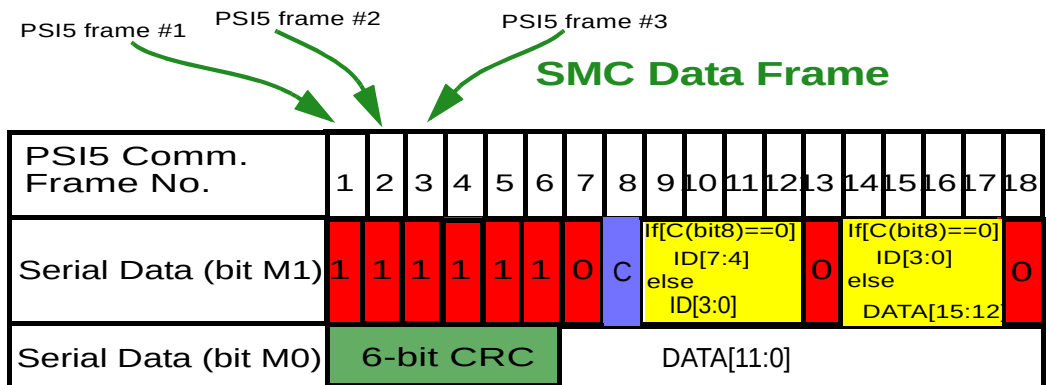
As the PSI5 module is targeted for powertrain applications, the data frame format differs from standard PSI5 frames for safety applications specified in version 1.3 of the PSI5 specifications. More specifically for supporting the version 2.0 powertrain substandard. The Powertrain data frame format and optional Serial Messaging Channel (SMC) frame format are shown in [Figure 1170](#).

Figure 1170. Receive data frame format

A) PSI5 Receive Data Frame Format (Power Train Applications)



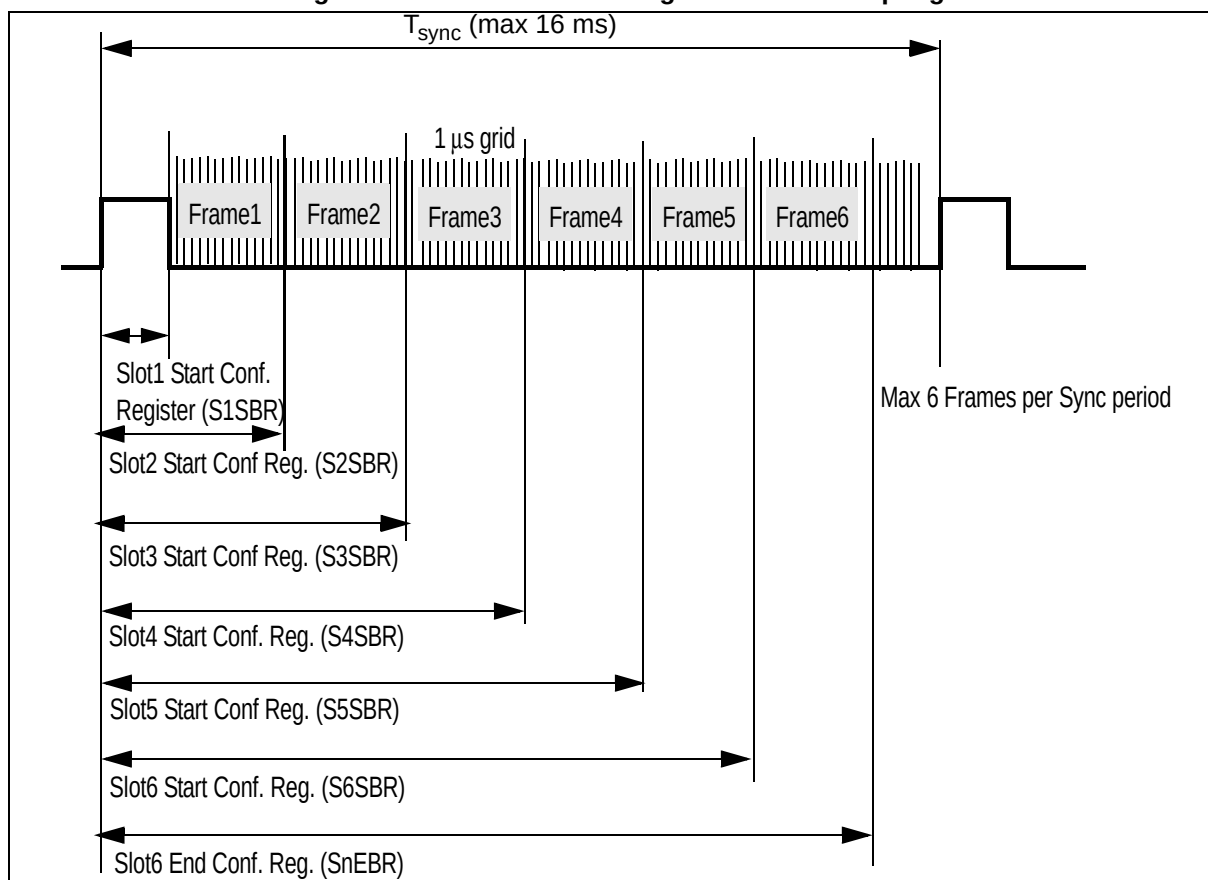
B) SMC Data Frame Format



Note: SMC frame encoded along with regular PSI5 data frames

53.4.2.2 Data Frame Configuration Control

Figure 1171. Data frame configuration and sampling



At the receive path, as many as six frame slots (reception windows) can be programmed with a start time of consecutive slots (slot1 to slot6) and end time or last slot, which can be configured through the Slot *n* Start/End Configuration Registers ([Section 53.3.2.28, Slot *n* Start Boundary Register \(PSI5_SnSBR\)](#) and [Section 53.3.2.29, Slot *n* End Boundary Register \(PSI5_SnEBR\)](#)) there are seven registers for six slots. The slot time value is with respect to timing sync pulse.

The following range is always maintained for T_{sync} (during programming of the GCR and CTPR registers) when timing (sync) pulses are generated through the Sync Pulse Generator.

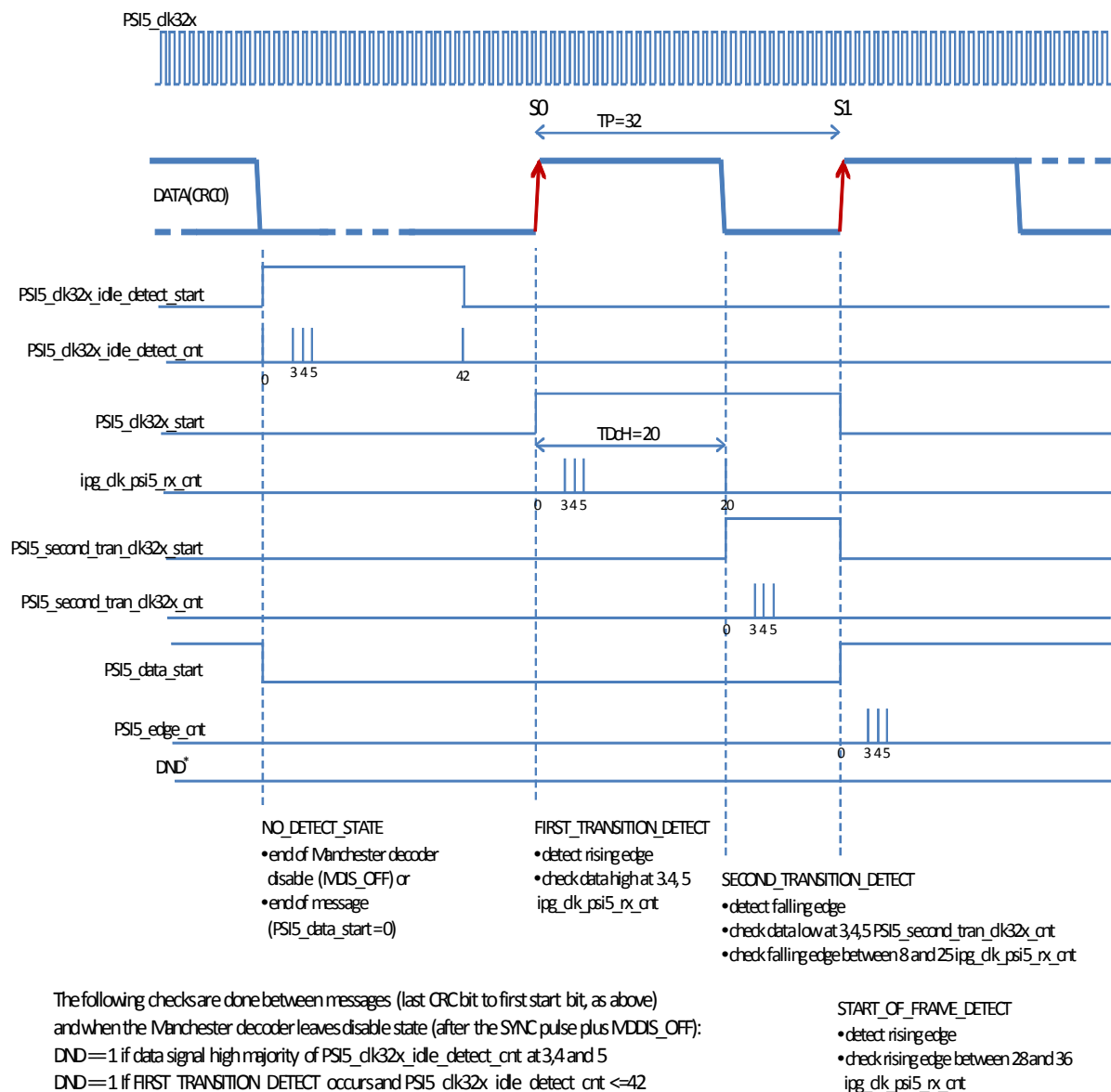
Equation 80 $T_{\text{sync}} < 16 \text{ ms}$

For the maximum defined six frame slots, six frame configuration registers (S1FCR, S2FCR, S3FCR, S4FCR, S5FCR, and S6FCR) are also available. In these registers it is possible to configure the payload region of the expected frames, in addition to other channel-specific parameters.

53.4.2.3 Data frame handling

[Figure 1172](#) shows the detection of the start bits and the various parameters for the same.

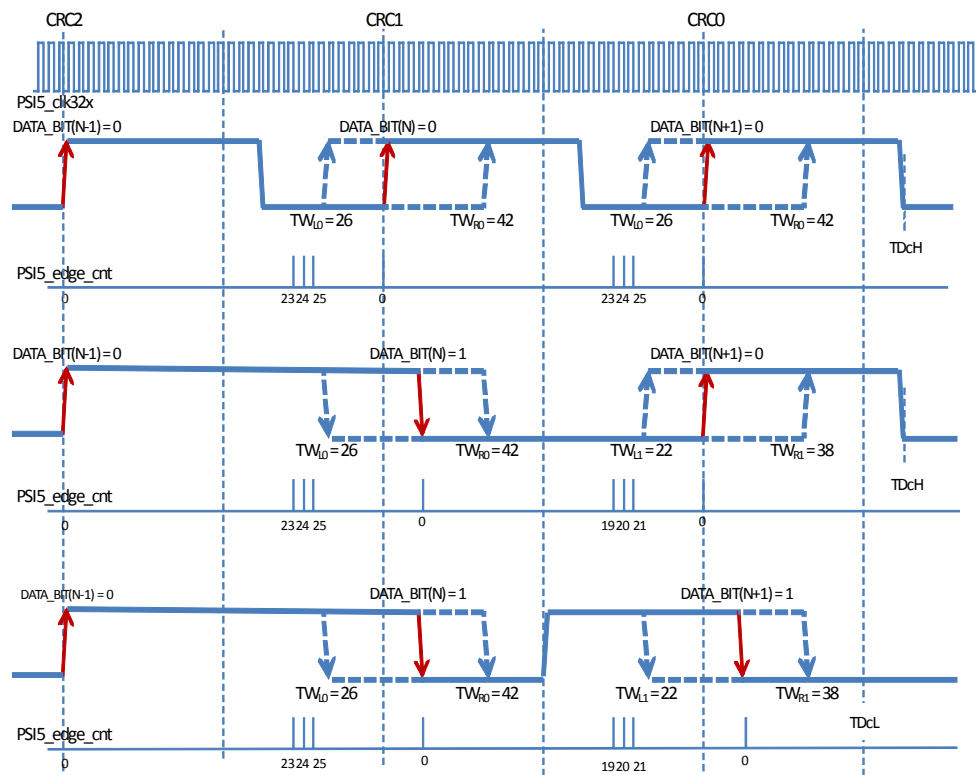
Figure 1172. Start Bit Detection



1. If DND is set to '1' because a message has overlapped with the Manchester decoder disable phase, then the Manchester decoder disable phase will be extended till the end of the first configured slot. DND will be cleared to '0' at the end of this first slot and NO_DETECT_STATE will be entered immediately. No idle detect will be carried out (i.e. PSI5_clk32x_idle_detect_cnt will not be started) to ensure that a message occurring within the first bit time of slot2 will be captured.
2. If DND is set to '1' because a first message (MSGn) has overlapped with a second message (MSGn+1), then MSGn+1 will be ignored i.e. the Manchester decoder will be disabled until the end of the slot in which MSGn+1 was expected. This will be the case whether the first transition detect of MSGn+1 occurs in the expected slot of MSGn or MSGn+1. At the end of the slot in which MSGn+1 was expected, DND will be cleared to '0' and NO_DETECT_STATE will be entered immediately. No idle detect will be carried out (i.e. PSI5_clk32x_idle_detect_cnt will not be started) to ensure that a message occurring within the first bit time of the slot in which the next message (MSGn+2) is expected will be captured.

Figure 1173 shows how the bits are differentiated into a “1” or a “0” based on the various margins and sample points.

Figure 1173. Bit Extraction



Start Sequence Measured: $T_p = 32$ and $TDCH = 20$

Calculations:

$TDCL = T_p - TDCH = 12$; $\Delta DC = TDCH - T_p/2 = 4$; $TMarginL = TDCL/2 = 6$; $TMarginH = TDCH/2 = 10$

If $DATA_BIT(N) = 0$, $TWL = TDCH + TMarginL = 26$; signal sampled at $TWL - 3$, -2 and -1

• if signal low, expect rising edge between $TWL0$ and $TWR0$ then $DATA_BIT(N+1) = 0$. $TWL0 = TDCH + TMarginL$, $TWR0 = T_p + TMarginH$.

• if signal high, expect falling edge between $TWL0$ and $TWR0$ then $DATA_BIT(N+1) = 1$. $TWL0 = TDCH + TMarginL$, $TWR0 = T_p + \Delta DC + TMarginL$.

If $DATA_BIT(N) = 1$, $TWL = TDCL + TMarginH = 22$; signal sampled at $TWL - 3$, -2 and -1

• if signal low, expect rising edge between $TWL1$ and $TWR1$ then $DATA_BIT(N+1) = 0$. $TWL1 = TDCL + TMarginH$, $TWR1 = T_p - \Delta DC + TMarginH$.

• if signal high, expect falling edge between $TWL1$ and $TWR1$ then $DATA_BIT(N+1) = 1$. $TWL1 = TDCL + TMarginH$, $TWR1 = T_p + TMarginL$.

PSI5_edge_cnt will be stopped (PSI5_data_start low) at $TDCH$ if $CRC0$ is logic 0 or $TDCL$ if $CRC0$ is logic 1.

53.4.2.3.1 Start of frame recognition

The start of frame recognition is as follows:

- With the Manchester decoder in NO_DETECT state, the first rising edge of the incoming data signal is captured and assumed to be the midpoint transition of the first start bit $S0$.
- As soon as the first rising edge ($S0$) on the bus is recognized, the 24-bit time stamp value is captured internally. On the rising edge of the $S1$ this value is stored in the data

receive register and available for processing. If a valid S1 edge is NOT recognized then this time stamp value is discarded and not stored in any register.

- The middle transition from the first start bit S0 to the second start bit S1 is assumed to be the bit duration for the rest of the bits.
- The time of the high period is measured using the 32* sample clock to enable analysis of the data signal duty cycle. The measured value is used in the data state to calculate the appropriate sample time of the first half-bit of the following data bits
- The second rising edge of the incoming data (S1) is captured and assumed to be the mid-point transition from the second start bit S1 if it appears in the allowed range of samples of the 32x sample clock.
- The time between the first and second rising edge is measured using the 32x sample clock to enable analysis of the sensor clock tolerance. The measured value is used together with the calculated duty cycle to set the window for capturing the mid-point bit transition the following data bits.
- [Figure 1172](#) and [Figure 1173](#) show the start bit and the data bit extraction mechanisms respectively, in the Manchester decoder.

53.4.2.3.2 Data state

In the data state, the Manchester decoder samples the number of data bits corresponding to the configured length of the data payload region and 3-bit CRC or 1-bit parity.

53.4.2.3.3 Message diagnostic checks

During the data state, electrical correctness of each bit in regard of Manchester encoding is checked. If an incorrect bit is detected, a 0 is written into the data payload region. Following completion of reception, correctness of the entire message is checked with the CRC and timing of the message reception.

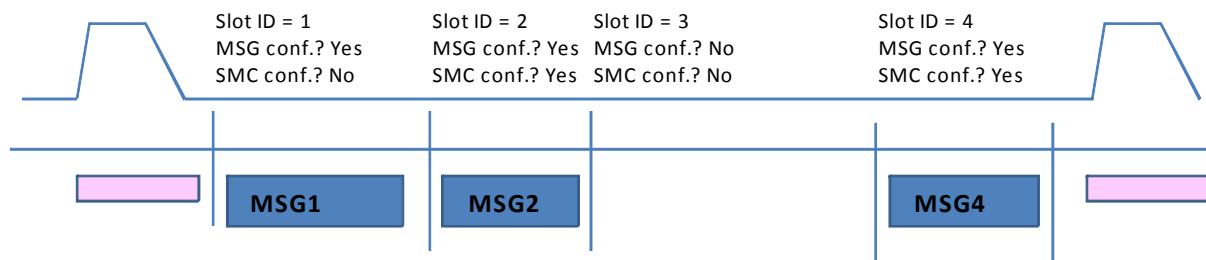
- CRC is recalculated for the data and compared with received CRC bits. The CRC comparison result is stored as a status bit, denoted as the C field in the receive register and the RAM register. Irrespective of the value of the "C" bit the, the frame is always stored in the RAM register and is never discarded.
- If at least one bit is electrically faulty (apart from M0M1), then the bit field E is set to 1 and stored as status bit in the receive register and the RAM register.
- The EM bit is set to 1 if any of M0 or M1 bits are electrically faulty. Note that the E and the EM bits can be set only when a valid start sequence has been detected.
- Set status bit field T in the receive register to 1 if the frame received is in the wrong slot (boundary violations). This bit is NOT set for asynchronous modes.
- Set status bit F if no frame is received in defined frame slot. This bit is NOT set for asynchronous modes.

53.4.2.3.4 The incoming frames after the sync pulse are to be numbered from 1 to 6 (max). This is done by a 3-bit slot counter, which is incremented automatically after the end of each slot irrespective of whether a valid message is received or not. This 3-bit value is stored in the receive register and the RAM registers as the "slot_counter" field. **T-bit Handling**

The following paragraphs describe how the IP handles error cases that generate T bit errors. A number of examples are shown, including the error free case, which is used to explain the reception process.

53.4.2.3.4.1 Error Free Case

Figure 1174. Error Free Case



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

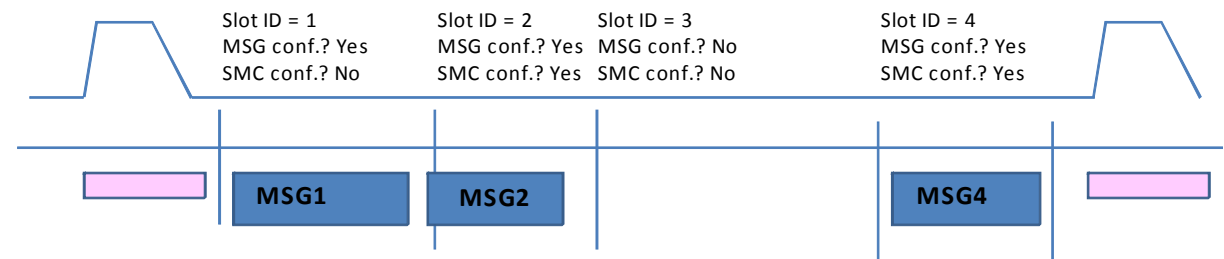
In this case, all messages are received in the correct slots.

- As soon as the Manchester decoder block is enabled (after the SYNC pulse has finished and any configured disable offset has expired, pink region) the IP checks for the first configured message and starts monitoring the output of the PSI5 transceiver.
- The IP identifies a correct start of message sequence ($S0, S1 == '00'$), and captures the slot counter when the first rising edge of the start sequence is detected. This first rising edge detected is the mid-point transition of the Manchester encoded S0 bit of the message start sequence
- The expected number of bits configured in the message is received and decoded. Any error in the Manchester encoded bits of the message E bit is set - unless the error is in one of the first two bits and SMC message is configured, in which case the EM bit is set.
- After the final bit is received, a check is made that the message CRC or parity is correct. If the CRC or parity check is not correct, C bit is set for the message.
- Once the CRC checks have been made, the received message — along with the status bits, slot counter in which the start sequence was received, the timestamp of the message, the message data and CRC or parity is saved to the channel message FIFO.
- As soon as the received message has been stored, the idle state of the bus is verified and monitoring of the transceiver output restarts to check for the next configured message
- If the serial message channel (SMC) is configured, the first two data bits are extracted from a sequence of 7 PSI5 messages as long as no electrical error is detected in those bits (EM error). If the correct start sequence for the SMC message is detected on M1 (6 logical 1s and a logical 0) in those 7 messages, M0 and M1 from the following 11 messages are also extracted and saved in the SMC buffer allocated to the slot in which

the PSI5 message is received (in total, 18 PSI5 messages are needed to build one SMC message). When the SMC has been received, the CRC is checked and the status of the check is saved in the SMC buffer. If an EM error occurs after a start sequence has been detected and before the end of the message is reached, M0 and M1 are both saved as 0 and the rest of the message is received, at which point the CRC is checked as normal.

53.4.2.3.4.2 Message Too Early

Figure 1175. Message Too Early



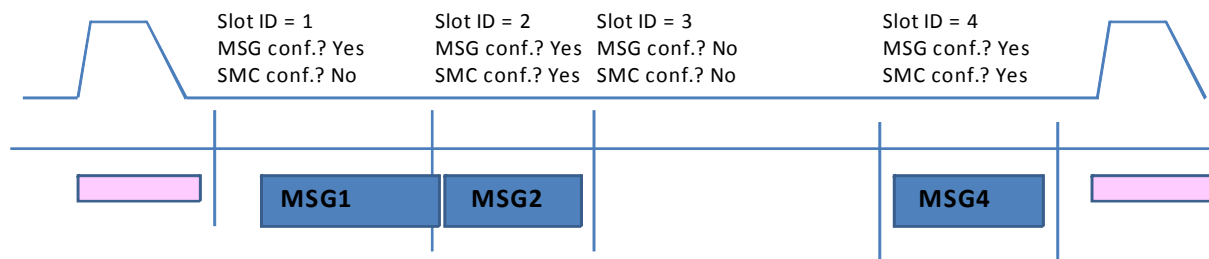
Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	1	Yes	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, MSG2 reception starts early, before slot 2 boundary is crossed, and so the slot counter is captured as “1” and because the message starts in the wrong slot, the T bit is set. If no other errors are detected during reception, the assumption is that the correct MSG2 has been received but not entirely in it’s correct slot. The status of the message will include the T bit set and slot counter will be 1, so software can interpret that MSG2 was too early.

The SMC message is received correctly.

53.4.2.3.4.3 Message Too Late

Figure 1176. Message Too Late



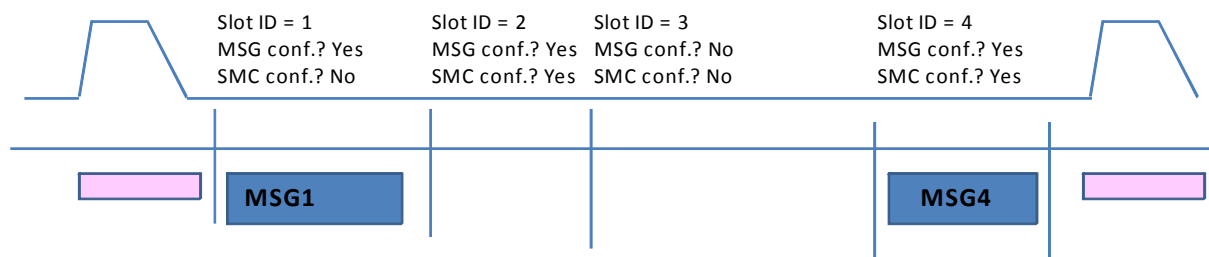
Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	Yes	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, MSG1 reception starts later than expected and ends after slot 2 boundary is crossed. The slot counter is captured as “1” and because the message ends in the wrong slot, the T bit is set. If no other errors are detected during reception, the assumption is that the correct MSG1 has been received but not entirely in it’s correct slot. The status of the message will include the T bit set and slot counter will be “1”, so software can interpret that MSG1 was late.

The SMC message is received correctly.

53.4.2.3.4.4 Message Missing

Figure 1177. Message Missing



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	Yes	No	No	2	0/0
MSG4	4	No	No	No	No	4	M0/M1

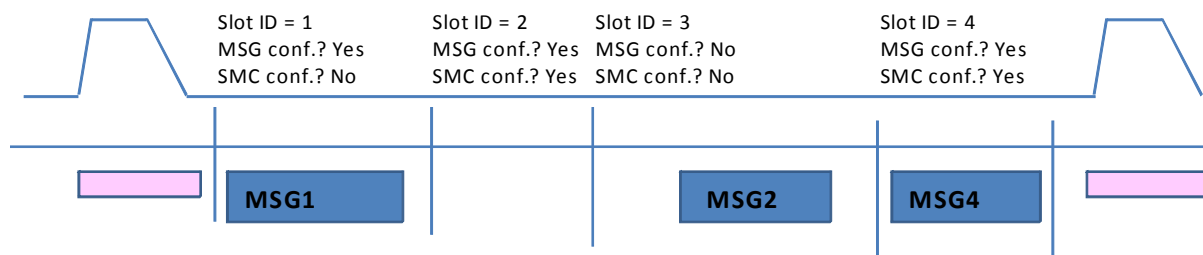
In this scenario, MSG2 is missing.

At the end of slot 2, the IP writes message buffer location 2 (MSG2 location) with slot counter 2 and the F bit set. The data and timestamp regions are filled with 0s.

The IP waits until the start of the next configured slot (slot 4) to update the SMC buffer 2 with "00". If slot 4 was not configured, the SMC for the missing MSG2 would be updated at the end of the final slot.

53.4.2.3.4.5 Message Much Too Late (in unconfigured slot)

Figure 1178. Message Much Too Late (in unconfigured slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2a	2	No	Yes	No	No	2	No update
MSG2b	3	Yes	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, MSG2 reception doesn't start in its own slot, and the F bit is set for message buffer location 2 (MSG2a).

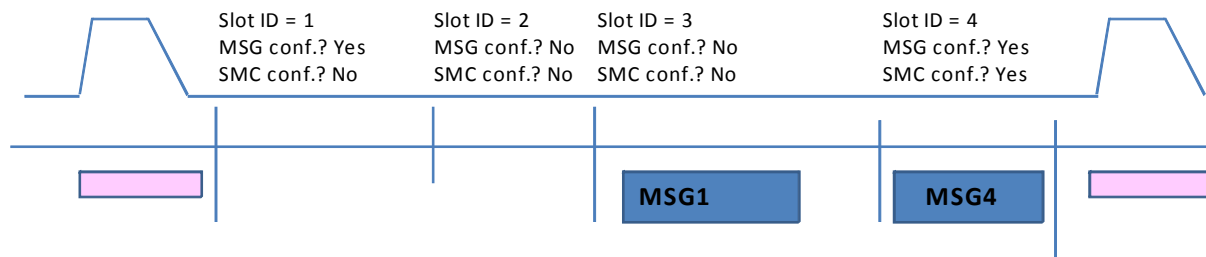
Because slot 3 is unconfigured, the IP continues to wait for MSG2 in slot 3 and does not update the SMC message buffer 2. The IP recognizes the MSG2 start condition in the normal way and updates PSI5 message buffer 3 (MSG2b), capturing slot 3 as the start slot counter and setting T bit, so that the software can recognize that MSG2 has come in slot 3.

When MSG2 has been received, SMC message buffer 2 is updated at the end of slot 3. MSG4 is received normally.

Note: .Any scenario where an expected message arrives in an unconfigured slot, leads to an issue when DMA access is used to read the PSI5 message buffer. This is because instead of the expected three entries in the PSI5 message buffer, there will be four, and assuming the DMA watermark is set for the expected three messages, the DMA request will be activated before MSG4 has been received.

In order to overcome this, if DMA access is required, no slots can be left unconfigured and the DMA watermark must be set to the number of slots in the SYNC period. In that way, the DMA will always be synchronised, but depending on how the 'empty' slot is configured, when a message is too late as described here, the result can be that the late message and its associated SMC content is lost.

53.4.2.3.4.6 Message Much Too Late (in unconfigured slot)

Figure 1179. Message Much Too Late (in unconfigured slot)

Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1a	1	No	Yes	No	No	No	No
MSG1b	3	Yes	No	No	No	No	No
MSG4	4	No	No	No	No	4	M0/M1

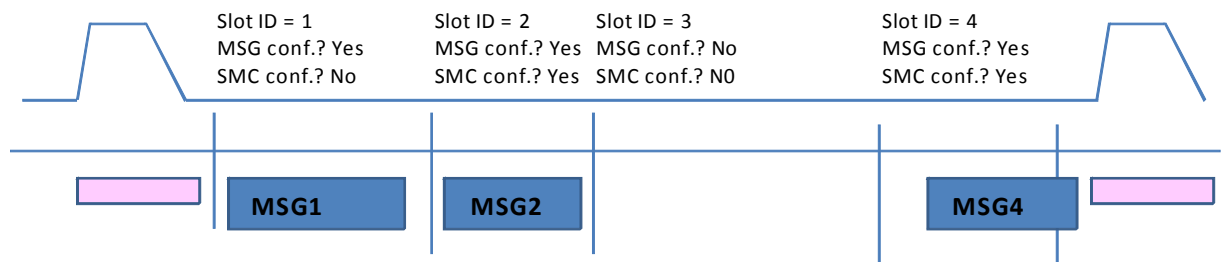
In this scenario, MSG1 is received much too late, in the second of two unconfigured slots.

When MSG1 does not arrive in slot 1, MSG1a is saved in the message buffer with F bit set. The IP waits throughout the two unconfigured slots and is received in slot 3 with slot counter 3. Because MSG1b is received completely in slot 3 it is saved in the message buffer with T bit set.

MSG4 is received normally.

53.4.2.3.4.7 Message Too Late (in last slot)

Figure 1180. Message Too Late (in last slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	Yes	No	No	No	4	M0/M1

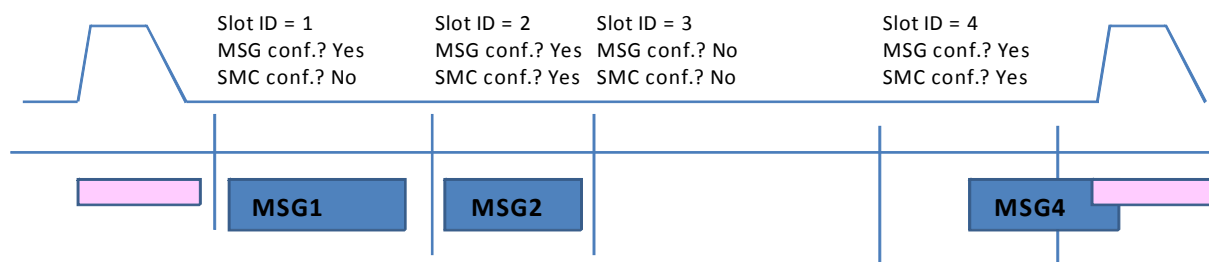
In this scenario, the last message is received late.

MSG1 and MSG2 are received error free.

MSG4 is late with the start sequence identified in the correct slot 4, with slot counter 4 captured. The end of the message occurs after the end of the final slot, but before the Manchester decoder is disabled by the SYNC pulse on the bus. The message is received correctly with the T bit set.

53.4.2.3.4.8 Message Too Late (in last slot)

Figure 1181. Message Too Late (in last slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	Yes	No	Yes	Yes	4	M0/M1 or 0/0

In this scenario, the last message is received late in the last slot.

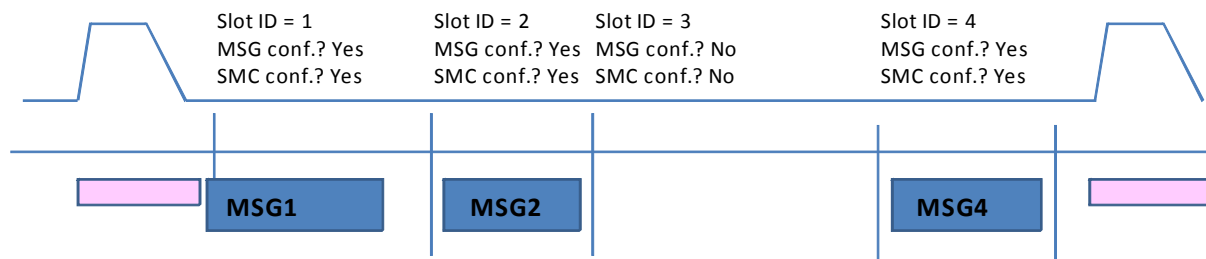
MSG1 and MSG2 are received error free.

MSG4 is late with the start sequence identified in the correct slot 4, with slot counter 4 captured. The end of the message occurs after the Manchester decoder is disabled by the SYNC pulse on the bus, so the last part of the message cannot be received correctly. Nevertheless, the message reception is completed and the message is stored in the message buffer with the T bit set and 0 written in the buffer for all bits received while the Manchester decoder is disabled. This leads to E and C bit errors.

In an extreme case, it could also lead to EM bit errors, in which case the SMC buffer will also be written with (0,0) otherwise, the correct bits are written to the SMC buffer.

53.4.2.3.4.9 Message Too Early (in first slot)

Figure 1182. Message Too Early (in first slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	0	Yes	No	No	No	1	M0/M1
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, the first message is received early.

As soon as the Manchester decoder disable phase has ended, the IP verifies the idle condition on the signal line and then looks for the first message start sequence. MSG1 start sequence is received before the start of slot 1 with slot counter 0 captured. MSG1 is then received normally and saved with slot counter 0 and T bit set.

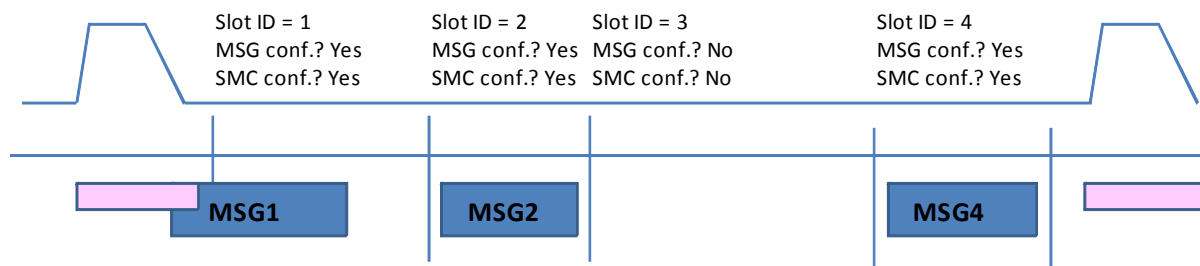
SMC bits M0/M1 are saved to SMC buffer 1.

MSG2 and MSG4 are received normally.

Note: *.This scenario can be avoided if the start of slot 1 is configured to coincide with the end of the Manchester decoder disable phase.*

53.4.2.3.4.10 Message Too Early (in first slot and overlaps Manchester Decoder disable)

Figure 1183. Message Too Early (in first slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	Yes	No	No	1	0/0
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

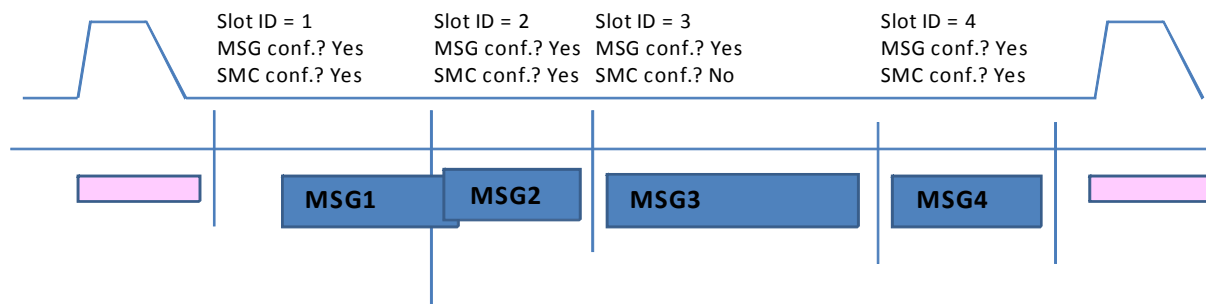
In this scenario, the first message is received early and overlaps with the Manchester decoder disable phase.

As soon as the Manchester decoder disable phase has ended, the IP starts looking for “idle condition” on the signal line by monitoring it for 42 counts of the sample clock. If the signal line is found to be initially high or if a valid transition occurs in the time frame of 42 samples of the sample clock, the Manchester decoder stops decoding until the end of slot 1. Decoding will restart at the beginning of slot 2.

MSG1 is assumed to be lost and the “F” bit is set for that message with the Slot Counter as “1”. If MSG1 SMC is configured then SMC is saved with “00” appended for the M0/M1 bits.

53.4.2.3.4.11 Message Too Late (overlaps next message)

Figure 1184. Message Too Late (overlaps next message)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	Yes	No	Yes	Yes	1	M0/M1
MSG2	2	No	Yes	No	No	2	0/0
MSG3	3	No	No	No	No	No	No
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, MSG1 is received too late and overlaps with MSG2.

MSG1 start is recognized in slot 1. The last bits of MSG1 are disrupted by the start of MSG2 leading to likely E and C bit errors.

At the end of MSG1, idle time is found to be violated by either the signal line being high as soon as NO_DETECT_STATE is entered or FIRST_TRANSITION_DETECT occurs within 42 counts of the idle detect sample clock. The Manchester decoder stops decoding until the end of slot 2. Decoding is restarted at the start of slot3. MSG2 is not received and MSG2 location in the memory will be saved with the F-bit set. The SMC buffer 2 will be updated with 00.

MSG3 and MSG4 are received correctly in their own slots.

53.4.2.3.5 Data Frame Storage

Once the data frame is received and all required status fields updated (for example, Time stamp, C, E, EM, T, F and Frame Counter), this 64-bit value in the data receive register is moved to the RAM Register (PMB).

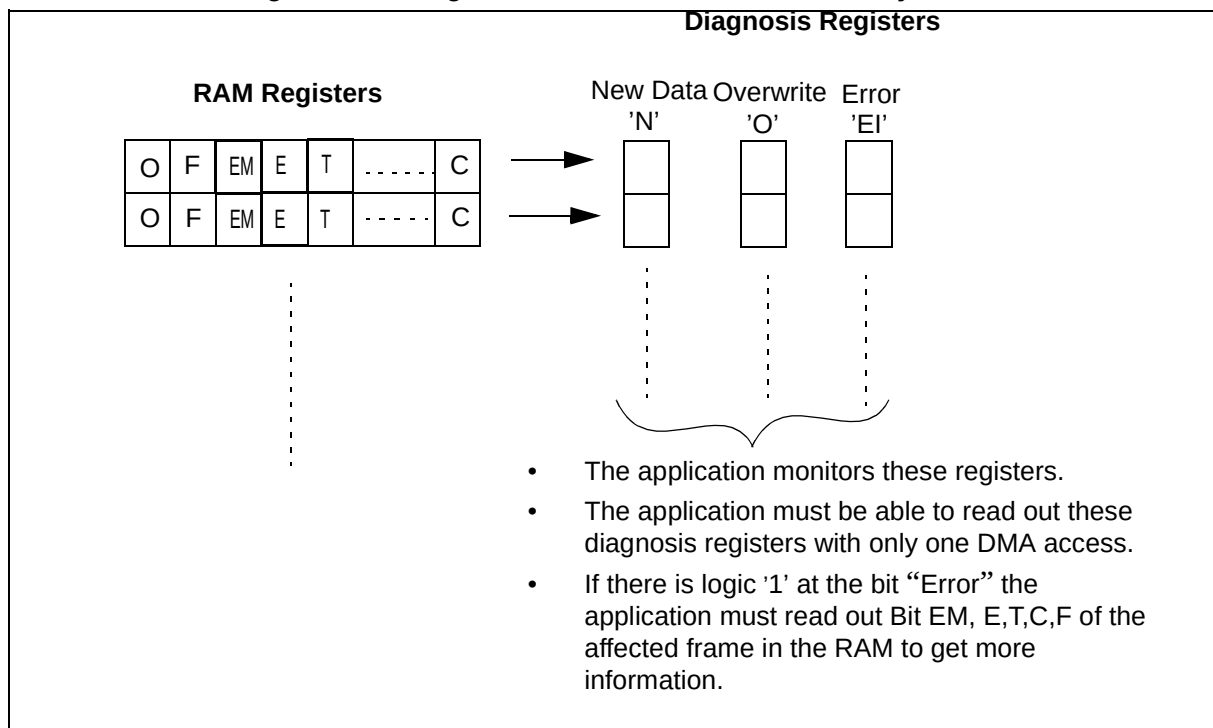
- This RAM buffer is a 32 (depth) × 64 (frame size) register array with DMA transfer capability (both single or burst).
- Also, the individual RAM locations can be accessed directly by the microcontroller core any time.
- Care needs to be taken that receive data is assigned to the sensor and frame slot unambiguously.

53.4.2.3.6 Message diagnostics

Apart from the regular status bits (mentioned above), a few more diagnostic checks are done and stored as status bits along with each data frame location in RAM. These are:

- Diagnosis registers are associated with each sensor data RAM buffer to indicate
 - RAM buffer has been filled with data.
 - RAM buffer has been overwritten before data previously stored was read.
 - Data written to buffer includes an error bit (E, EM, C, T, or F, configurable).
- The diagnosis registers are able to be read with a single DMA access.
- After successfully completing a read access, these registers are reset by hardware. This is done by mapping these diagnosis bits byte-wise so that one byte access clears a single bit without affecting other bits.

Figure 1185. Diagnosis Information attached to every frame



Note: See [Section 53.8.3, Implementation of EI Diagnostic Register](#) for more detail.

53.4.2.3.7 SMC message extraction

The optional messaging channel M0, M1 (shown in [Figure 1170](#)) is used by sensors to send an SMC frame along with the normal PSI5 frame. It takes 18 PSI5 frames to send one SMC frame.

The extraction of SMC bits is done for each slot separately and stored separately into the SMC frame receive buffers. The size of the SMC frame receive buffers is 6 (no of slots) × 32 (width of SMC frame, as defined below)

The extraction logic works as follows:

- Search for SMC start sequence 111111 followed by 0 on M1 and keep storing last 6 bits {CRC} on M0.
- If the start sequence is detected it continues with further extraction and storage of M1 and M0.
 - Store the extracted frame on the corresponding slot buffer.
 - Move the SMC frame to corresponding SFR[i] register period. Once complete, the SMC frame is extracted and generates an interrupt, so that it can be read by the CPU.
 - If the start sequence is not detected, it continues with the search.

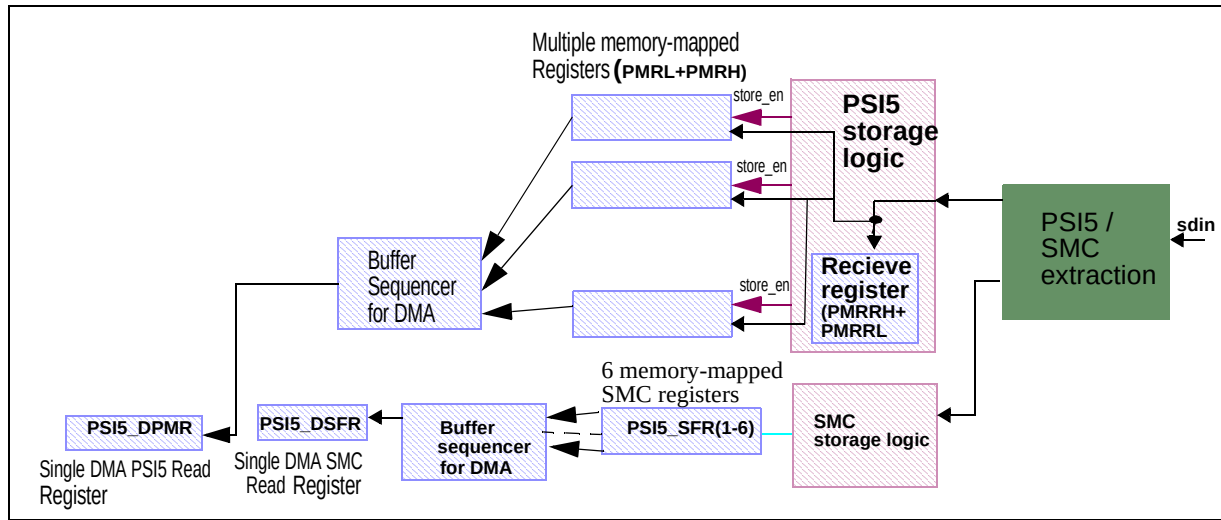
The SMC frame is extracted in hardware and stored separately as memory-mapped registers. The size of the SMC frame receive buffers is 6 (no of slots) × 32 (width of SMC frame).

53.4.2.4 Message Read Logic

PSI5 supports two kinds of read access to received sensor data.

- Read access through memory-mapped RAM registers. These registers can be read by the CPU any time or, specifically, when the concerned interrupts are raised. Applicable for both the SMC and the PSI5 messages.
- Read access through DMA request. Here the DMA sequencer treats the memory-mapped registers as a block and reads them sequentially. Note that even if the DMA is used, the CPU can read the RAM registers any time by addressing them. This is applicable for both the SMC and the PSI5 messages. When reading the PSI5 messages, it is possible to read only the PSI5 messages or PSI5 + diagnostic registers.

Figure 1186. PSI5 and SMC Message Read Logic Block



53.4.2.5 CRC Recalculation

Error detection is realized by a single bit even parity (recommended for 10 or fewer bits) or a 3-bit CRC (recommended for long data words). The transmission error detection is selectable (by frame configuration register) as:

- 1-bit even parity
- 3-bit CRC

The generator polynomial of the CRC is $g(x)=1+x+x^3$ with a binary CRC initialization value of 111. The transmitter extends the data bits by three 0s (as MSBs). This augmented data word is fed (LSB first) into the shift registers of the CRC check. Start bits are ignored in this check. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the buffer registers contain the CRC checksum. These three check bits are transmitted in reverse order (MSB first: C2, C1, C0).

Figure 1187. CRC calculation scheme

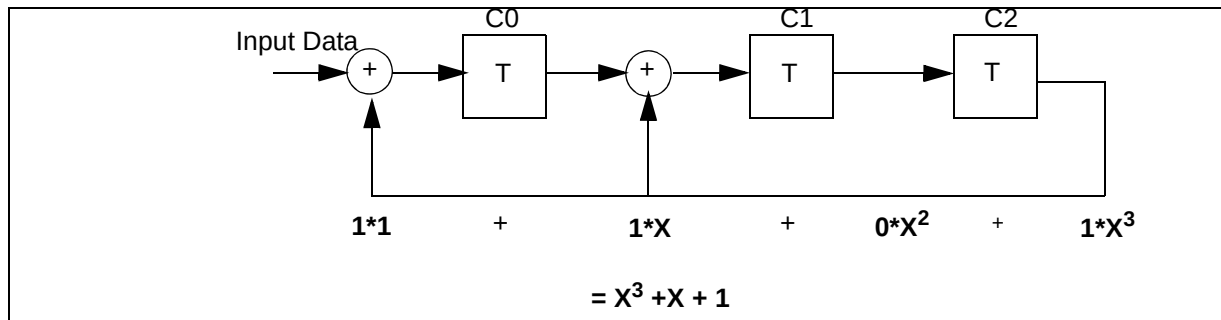


Table 1174. CRC Calculation (at transmitter side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	0	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	1	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	1	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	1

Table 1175. Transmitted and Received Sensor Data

Transmitted Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0
Received Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

Table 1176. CRC Re-Calculation (At Receiver Side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	1	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	0	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	0

Syndrome = 000 → Transmission successful

The correct transmission is checked by the PSI5 receiver by comparing the syndrome (content of the shift register after reception of both data bits and CRC bits) with the zero vector. For error free reception, all shift registers have to be zero.

53.4.3 ECU-to-sensor communication

Whereas the sensor-to-ECU communication is realized by current modulation, voltage modulation on the supply lines is used to transmit commands to the sensors. A predefined start sequence allows the sensors to recognize that a command is being transmitted during all synchronous operation modes.

53.4.3.1 Physical Layer

A logical 1/0 is represented by data sync pulse of variable length, a logical 1 by the sync pulse of width W1(programmable from 0 to 127 μ s) and logical '0' by sync pulse with W0{programmable from 0 to 127 μ s}.

53.4.3.2 Data Link Layer

53.4.3.3 Bit coding

The bit period is the cycle time as specified for the operation mode, for example, 500 μ s in the PSI5-P10P-500/3L mode. (For details about this topology, please refer [Section 53.1.3.3.1, PSI5 topology](#).) The lower limit depends on the filter characteristics of the hardware circuitry used for the sync signal detection (subsequent sync signals might vary the internal $V_{ss, base}$ reference).

53.4.3.4 Data transmission

The IP supports the standard and the Non-Standard ECU to Sensor communication Formats. In the Non-Standard mode of communication it can be used to send any frame size programmable from 1-64 bits. The IP supports the PSI5 ECU to Sensor Communication formats as briefed below in [Figure 1188](#) and the text that follows it.

Figure 1188. Standard Frame Formats Supported

Frame 1, Short

Start				SAdr			FC				CRC			
0	1	0	S	A0	A1	A2	S	F0	F1	F2	S	C2	C1	C0

N° Bits: 15 (including start+stuff)

 Synchronization Bit [1]

Frame 2, Long (4-Bit Data Nibbles)

Start				SAdr			FC			RAdr						Data				CRC								
0	1	0	S	A0	A1	A2	S	F0	F1	F2	S	0	1	0	S	X3	X4	X5	S	D0	D1	D2	S	D3	C2	C1	S	C0

N° Bits: 29 (including start + stuff)

Frame 2, Long (8-Bit Data Word)

Start				SAdr			FC			RAdr		Data								CRC								
0	1	0	S	A0	A1	A2	S	F0	F1	F2	S	X0	X1	D0	S	D1	D2	D3	S	D4	D5	D6	S	D7	C2	C1	S	C0

N° Bits: 29 (including start+stuff)

Frame 3, XLong

Start				SAdr			FC			RAdr			Data			CRC					
0	1	0	S	A0	A1	A2	S	F0	F1	F2	S	X0 - X7 + Sync Bits			D0 - D7 + Sync Bits			C2	C1	S	C0

N° Bits: 37 (including start+stuff)

 Synchronization Bit [0]

 Configuration Bit
 [1] Mapping of frames 1-3
 [0] all other applications

Frame 4, XXLong

Start									SAdr				Data										CRC							
S	1	1	1	1	1	1	1	S	A0	A1	A2	C	D0	D1	S	D2	...	D7	S	D8 - D19 + Sync Bits				S	C0	C1	C2	C3	C4	C5

N° Bits: 43 (including start+stuff)

- Short:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- "010", S, "A0, A1, A2", S, "F0, F1, F2", S, "C2, C1, C0" where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is the sensor

address, F0-F2 are the functional codes and "C2-C0" is the CRC. Please refer [Figure 1188](#) for the frame structure.

- The CRC polynomial used is $x^3 + x + 1$ with a binary initialization value of "111(C2:C0)".
- The transmitter extends the data bits by three zeros (as MSBs).
- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
- When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[5],000) with DPR[0] being fed first in the CRC calculator.
- **Long(4bit Data Nibble):** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- "010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, X2", S, "X3, X4, X5", S, "D0, D1, D2", S, "D3, C2, C1", S, "C0" where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes, "X0-X5" is the Range Address, "D0-D3" is the data and "C2-C0" is the CRC. Please refer [Figure 1188](#) for the frame structure.
 - The CRC polynomial used is $x^3 + x + 1$ with a binary initialization value of "111(C2:C0)".
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0..X5,D0...D3,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
 - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[15],000) with DPR[0] being fed first in the CRC calculator.
- **Long(8bit Data Word):** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - "010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, D0", S, "D1, D2, D3", S, "D4, D4, D6", S, "D7, C2, C1", S, "C0" where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes, "X0-X1" is the Range Address, "D0-D7" is the data and "C2-C0" is the CRC. Please refer [Figure 1188](#) for the frame structure.
 - The CRC polynomial used is $x^3 + x + 1$ with a binary initialization value of "111(C2:C0)".
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0..X1,D0...D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
 - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[15],000) with DPR[0] being fed first in the CRC calculator.
- **X-Long:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - "010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, X2", S, "X3, X4, X5", S, "X6, X7, D0", S, "D1, D2, D3", S, "D4, D5, D6", S, "D7, C2, C1", S, "C0" where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is

the sensor address, F0-F2 are the functional codes, "X0-X5" is the Range Address, "D0-D3" is the data and "C2-C0" is the CRC. Please refer [Figure 1188](#) for the frame structure.

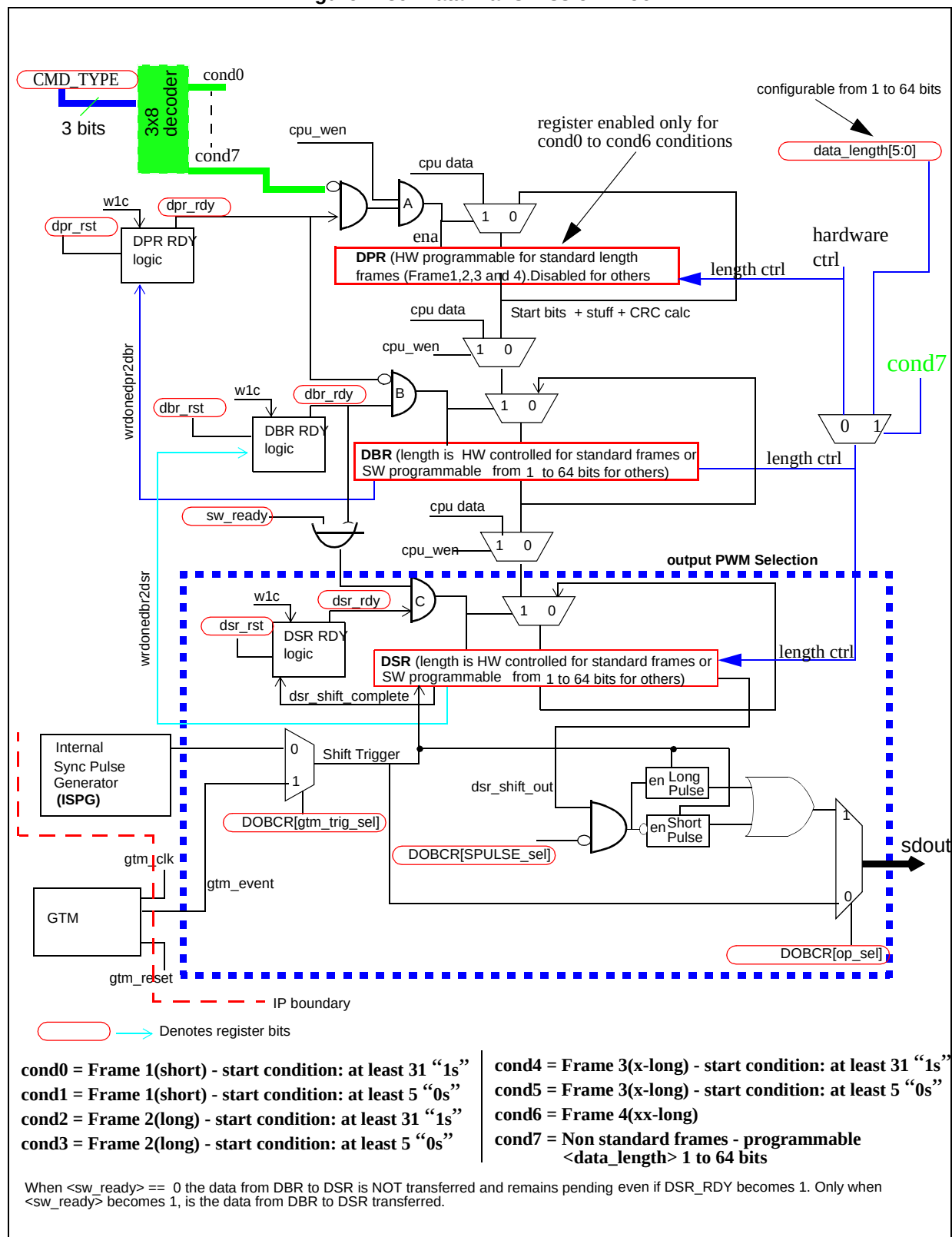
- The CRC polynomial used is $x^3 + x + 1$ with a binary initialization value of "111(C2:C0)".
- The transmitter extends the data bits by three zeros (as MSBs).
- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0...X7,D0...D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
- When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[21],000) with DPR[0] being fed first in the CRC calculator.
- **XX-Long:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C5 comes Last) Data Stream - "011111110", "A0, A1, A2", "C", "D0, D1", S, "D2, D3, D4" D5, D6, D7, S, "D8, D9, D10, D11, D12, D13", S,"D14, D15, D16, D17, D18, D19", S, "C0, C1, C2, C3, C4, C5" where "011111110" is the start sequence "S" is the stuff bit(0), A0-A2 is the sensor address, "C" is the configuration bit, "D0-D19" is the data and "C0-C5" is the CRC. Please refer [Figure 1188](#) for the frame structure.
 - The CRC polynomial used is $x^6 + x^4 + x^3 + 1$ with a binary CRC initialization value 010101(C5:C0).
 - The transmitter extends the data bits by six zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,D0..D19,"000000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
 - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[23],000000) with DPR[0] being fed first in the CRC calculator.

53.4.3.4.1 Transmission Data Framing

The output data transmission block is constructed of seven sub-blocks:

- Data preparation register
- Buffer register
- Data shift register
- Periodic Sync Pulse Trigger Generator
- "GTM driven" sync pulse trigger input (GTM external to PSI5)
- Pulse length modulation block
- OR gate

Figure 1189. Data Transmission Block



53.4.3.4.2 Data Preparation Register (64-bit register, DPRH + DPRL, DPRH is fixed at 0)

- Up to 24 bits of this register can be written by the CPU. These register bits contain the ECU to sensor command that needs to be transmitted after being appended with the CRC, start bits and stuff bits. The hardware automatically calculates the CRC, adds the start and the stuff bits, before transferring the command to DBR. When `DPR_RDY == 1`, the data to be transmitted to sensors is written to this register via system bus with order data bit to be transmitted first at LSB and furthest on MSB. This register can only be used for writing standard length data frames. Data to this register can be written only when `PSI5_DOBCR[CMD_TYPE] != 7`. When `PSI5_DOBCR[CMD_TYPE] == 7` then `DPR_RDY` remains at "0" and the commands cannot be written to this register. Bits writable in different settings of the `PSI5_DOBCR[CMD_TYPE]` can be seen in [Section 53.3.2.36, Data Preparation Register Low \(PSI5_DPRL\)](#). Note that `PSI5_DPRH` is always = 0.
- Having written the data to the DPR, the `DPR_RDY` bit has to be cleared with a w1c. This indicates to the internal logic that the data in the DPR is valid and is to be processed.
- If `DPR_RDY = 0` and a new command is written to the DPR then the command is rejected and `GISR[IS_PROW]` is set.
- If the CRC or parity bit calculation is finished, then the data is shifted to the Buffer register with the start bits and the sync bits appended as well. This transfer is done to the Buffer register only if the Buffer register is ready for new data, which is indicated by `GISR[DBR_RDY] == 1`.
- As mentioned above, once this transfer is done, the `DPR_RDY` becomes 1 and a new command can be written to it.

53.4.3.4.3 Data Buffer Register (64-bit register)

- This register has two subregisters `DBRH` (higher register) and `DBRL` (lower register). Both of them are collectively referred to as `DBR`. `DBR` is of length 64 bits `DBR[63:0]`
- The bits accessible for functionality are dependent on the value of `PSI5_DOBCR[CMD_TYPE]`. For the writable bits corresponding to different setting of the `PSI5_DOBCR[CMD_TYPE]` field please refer bit description in `CMD_TYPE[2:0]`
- When `DPR_RDY == 0` and `DBR_RDY` goes to 1 then internal logic automatically transfers the data from the DPR (with CRC + stuff bits + sync bits) to the DBR. `DBR_RDY` goes low and `DPR_RDY` goes 1. In order to check the calculation of the DPR, register `DBR` can be read at this stage, i.e., when `DPR_RDY` becomes "1".
- If `DBR_RDY == 1` AND `DPR_RDY == 1`, then the data to be transmitted to sensors can be written to this register directly via system bus skipping preparation register in case application includes the start and sync bits and calculates the CRC/Parity itself. If the CPU writes to the DBR when `DBR_RDY == 1` but `DPR_RDY == 0` and

PSI5_DOBCR[CMD_TYPE] != 7, then the results can be unpredictable since the DPR has a pending data that needs to be transferred to DBR.

- If the CPU is writing to the DBR as mentioned above, then the DBR_RDY bit has to be cleared by a w1c. This indicates to the internal logic that the data in the DBR is valid and is to be used for the shift operation.
- If while DBR_RDY == 0, and a new command is directly written to the DBR then the command is rejected and GISR[IS_BROW] bit is set.
- If while DBR_RDY == 0 a new command HAS to be written in the DBR then DOBCR[DBR_RST] has to be written to 1. Writing a 1 to DBR_RST will reject the current command in the DBR and generate DBR_RDY = 1.
- If the Data Shift Register signals that it is ready for new data (PSI5_GISR[DSR_RDY] == 1), the data of the Buffer register is shifted into the data shift register provided PSI5_DOBCR[SW_READY] is 1.
- If PSI5_DOBCR[SW_READY] == 0 then the data from the DBR is NOT transferred to the DSR automatically even if the DSR_RDY goes as 1. In such a case the IP waits for the software to make the PSI5_DOBCR[SW_READY] bit as 1; only then does it transfer the contents to the DSR.

53.4.3.4.4 Data Shift Register (64-bit register)

- This register has two subregisters DSRH (higher register) and DSRL (lower register). Both of them are collectively referred to as DSR. DSR is of length 64 bits DSR[63:0]. This register has a 1-1 correspondence with the bits in the DBR register when this register is updated automatically by the hardware.
- The bits accessible for functionality are dependent on the value of PSI5_DOBCR[CMD_TYPE]. For the writable bits corresponding to different setting of the PSI5_DOBCR[CMD_TYPE] field please refer bit description in CMD_TYPE[2:0].
- When DBR_RDY == 0 and DSR_RDY goes to 1 then internal logic automatically transfers the data from the DBR to the DSR provided PSI5_DOBCR[SW_READY] == 1'b1, else this transfer remains pending until PSI5_DOBCR[SW_READY] is '0'. Once the transfer is complete, DSR_RDY goes low and DBR_RDY goes 1. DSR contents that would finally get transferred, can be read at this stage, i.e., when DBR_RDY becomes "1".
- Making PSI5_DOBCR[SW_READY] now as '0' will not interrupt the ongoing transfer in the DSR.
- If DPR_RDY == 1 AND DBR_RDY == 1 AND DSR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping Buffer register. If the CPU writes to the DSR when DSR_RDY == 1 but DBR_RDY == 0 then the results can be unpredictable since the DBR has a pending data that needs to be transferred to DSR.
- When PSI5_DOBCR[CMD_TYPE] == 7 and If DBR_RDY == 1 AND DSR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping Buffer register. If the CPU writes to the DSR when DSR_RDY ==

1 but `DBR_RDY == 0` then the results can be unpredictable since the DBR has a pending data that needs to be transferred to DSR.

- If the CPU is writing to the DSR as mentioned above then the `DSR_RDY` bit has to be cleared by a w1c. This indicates to the internal logic that the data in the DSR is valid and is to be used for the shift operation.
- If while `DSR_RDY == 0`, and a new command is directly written to the DSR then the command is rejected and `GISR[IS_DSROW]` is set.
- If while `DSR_RDY == 0` a new command HAS to be written in the DSR then `DOBCR[DSR_RST]` has to be written to 1. Writing a 1 to `DSR_RST` will reject the current command in the DSR and generate `DSR_RDY = 1`.
- This register is clocked by the ISPG pulses or the GTM pulses, as described in [Section 53.4.3.4.5, Sync Pulse Trigger Generation](#).
- Once the shift operation is complete in the DSR then `DSR_RDY` becomes 1.

53.4.3.4.5 Sync Pulse Trigger Generation

The sync pulse trigger can be used to generate the timing pulses or the data pulses.

The sync pulse can be generated based on triggers from two different sources:

- **ISPG (Internal Sync Pulse Generator):** Composed of CTPR, CIPR and Channel Target Counter (per channel). The internal sync pulse generator is clocked by either the GTM clock (max frequency is 3.9Mhz) or the internal 1 MHz clock. The Channel Target Counter operates with a 1 μ s resolution (when using the 1Mhz clock) or GTM clock resolution (when using the GTM clock). The ISPG is implemented to allow generation of internal periodic trigger. Please Refer [Figure 1196](#) for the ISPG structure and [Figure 1194](#) for the CTPR and CIPR explanation. The ISPG uses the Channel Target Counters (CTCs) for generating the sync pulse. The CTCs can be triggered globally (simultaneous for all channels) or locally (individual to each channel). The `CTC_GED` bit in the Global Control Register (GCR) ensures that all the CTCs start/stop simultaneously. The individual target values when the trigger pulse is generated are set in the Counter Target Pulse Register (CTPR) and the Channel Initial Pulse Register (CIPR). These registers are present for each channel. When staggering the channels, the `CTC_GED` bit should be used for providing a simultaneous trigger to the staggered channels. It is also possible for a subset of channels to work on the above arrangement while the remaining channels independently generate trigger pulses for which start/stop is controlled locally to the channels.
 - For each channel, it is possible to configure the trigger period by software using sync pulse generator control/configuration register and setting compare values in Counter Target Pulse Register(s). The Counter Initial Pulse Register (CIPR) can be used to provide the initial offset between various channels.
 - It is possible to deactivate each channel separately for the purpose of sync pulse generation.
 - The periodic pulses can be used to synchronize sensor-to-ECU communication (synchronization timing pulses) [State 1 of [Section 53.8.2, Valid states for integrated sync pulse generator](#)] or can be used as data pulse during ECU-to-sensor communication (through the DOBCR) [State 2 of [Section 53.8.2, Valid states for integrated sync pulse generator](#)].
 - When programming the CTPR, the Tsync period should be programmed ≥ 6 `SP_TS_CLK` cycles (For details of `SP_TS_CLK`, refer to Bit 23 of

Section 53.3.2.2, PSI5 Channel Control Register (PSI5_PCCR).

- **GTM event controlled:** Generation of sync trigger/pulse can be done based on events (for example, crank angle or software) through **general timer module (GTM)** output to the PSI5 interface.
 - Here the sync trigger pulses come from an external GTM module routed through a full XBAR, which is programmable at the system level.
 - The sync pulse programming (period, and so on) is done inside the GTM.
 - In [Figure 1189](#), if DOBCR[gtm_trig_sel] == 1, DOBCR[SPULSE_sel] == 0 and DOBCR[OP_sel] == 1 then the GTM trigger is used for data pulse generation (data pulses for ECU-to-sensor communication). [State 4 of [Section 53.8.2, Valid states for integrated sync pulse generator](#)]
 - In [Figure 1189](#), if DOBCR[gtm_trig_sel] == 1, DOBCR[SPULSE_sel] == 1 and DOBCR[OP_sel] == 1 the GTM trigger is used for timing pulse generation (synchronization pulse for sensor-to-ECU communication). [State 3 of [Section 53.8.2, Valid states for integrated sync pulse generator](#)]
 - In [Figure 1189](#), if DOBCR[gtm_trig_sel] == 1, DOBCR[SPULSE_sel] == "x" and DOBCR[OP_sel] == 0 the GTM trigger is passed directly to data sync port without pulse length modulation. In this case, the sync pulse length is defined by the GTM pulse length. [State 5 of [Section 53.8.2, Valid states for integrated sync pulse generator](#)]
- **Programming and control:** For achieving the various states of ISPG control or the GTM control, the 3 equivalent bits (GTM_TRIG_SEL, SPULSE_SEL and OP_SEL) belonging to the DOBCR, can be programmed, as shown in [Figure 1189](#) and [Figure 1190](#). The programming of various states mentioned above, with reference to the [Section 53.8.2, Valid states for integrated sync pulse generator](#) can be seen in [Table 1162](#) and also shown in [Figure 1197](#). The sending of the ECU to sensor commands, can be controlled by PSI5_GISR[DPR_RDY, DBR_RDY, DSR_RDY] bits and the PSI5_DOBCR register bits.

53.4.3.4.5.1 Timing Pulses

Timing pulses can also be produced without pulse width modulation also. These timing pulses can be generated through two sources:

- Periodic sync pulse from ISPG. In this case the synchro pulse from the internal sync pulse generator can be made available without any modulation. Referring [Figure 1190](#), in synchronous mode, if DOBCR[GTM_TRIG_SEL] == 0 and DOBCR[OP_SEL] == 0 then the internal sync pulse generator output can be made available without any modulation. The high time of this pulse is fixed at 4 μ s.
- Referring [Figure 1190](#), in synchronous mode, if DOBCR[GTM_TRIG_SEL] == 1 and DOBCR[OP_SEL] == 0, then the GTM trigger can be used for timing pulse generation without any Pulse Width Modulation.

53.4.3.4.5.2 Data pulse generation

Data pulses are produced by the Pulse Length Modulation Block based on triggers from one of the two selectable sources, the **ISPG** or the **GTM**, during ECU-to-sensor communication.

The two sync sources are selected by software programming of the Pulse Length Control/Configuration Register.

- A logical 1 in the Data Output Register LSB is coded as data sync pulse of width W1 {up to 127 μ s}.
- A logical 0 in the Data Output Register LSB is coded as data sync pulse of width W0 {up to 127 μ s}.
- The two widths are programmable in the Pulse Length Control/Configuration Register.
- The Data Shift Register is shifted right every time a bit is transmitted.

Figure 1189 shows the data transmission block.

Figure 1190. ECU-to-sensor communication registers (implementation of Figure 1189)

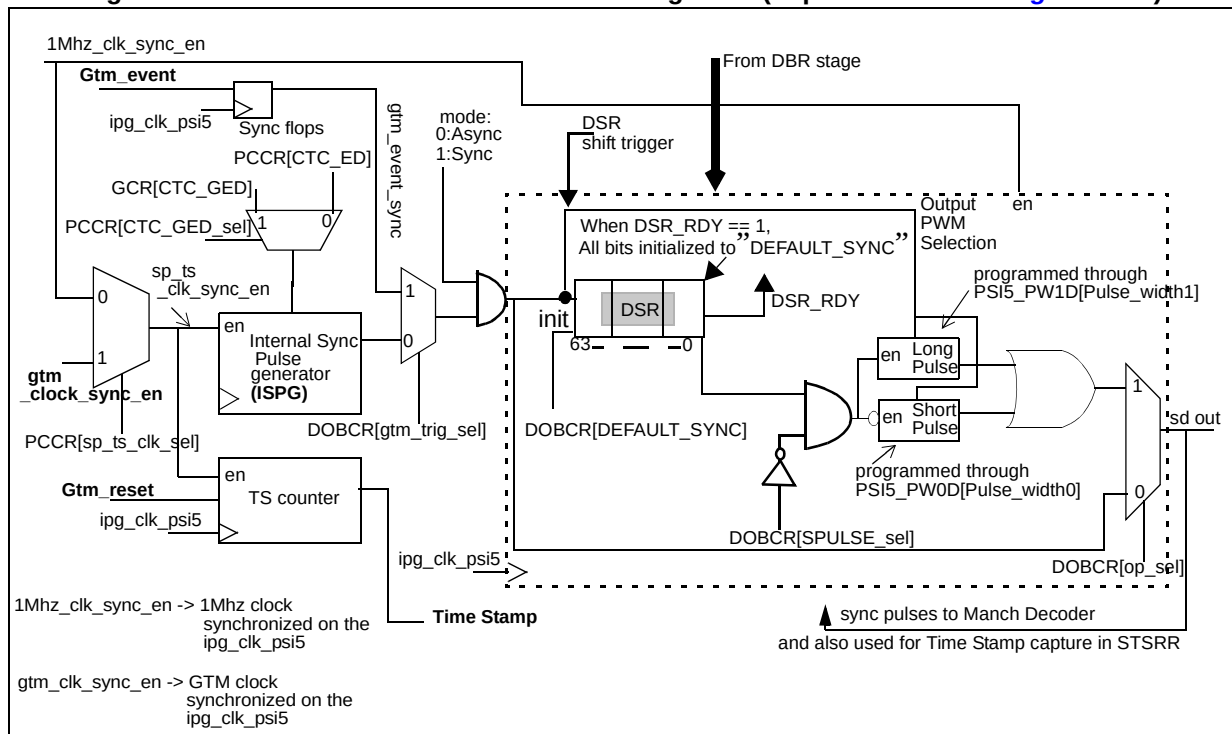


Figure 1190 details the implementation of Figure 1189 and shows how the various bits mentioned in Section 53.3.2.30, *Data Output Block Configuration Register (PSI5_DOBCR)* control the behavior of the Output Command path. Figure 1190 also shows the various clock sources to the Internal Sync Pulse Generator and the Time Stamp counter.

53.4.3.5 Response to ECU-to-sensor frames

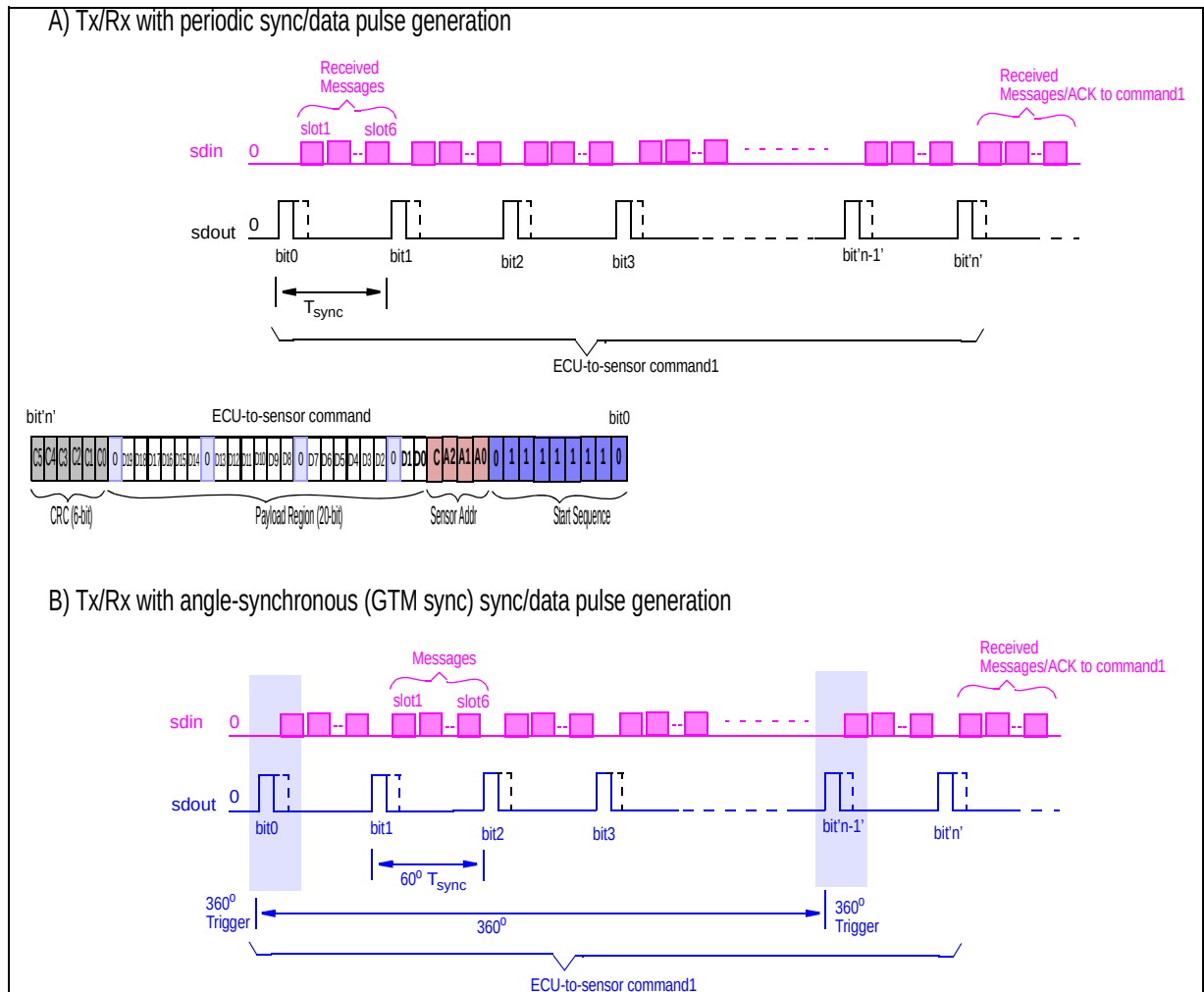
The response to an ECU-to-sensor frame can be optionally transmitted in two ways:

- Out-of range values of data frame (as fast response coded as normal PSI5 payload region)
- Serial messaging channel (verification of the response frames must be handled by the software)

53.4.4 Tx/Rx scenario

The diagram below shows the communication scenario in synchronous modes. For asynchronous mode, sync pulses are not generated and sensor transmits data asynchronously.

Figure 1191. Synchronous mode communication scenarios



53.4.5 Interrupt handling

Configurable interrupts based on events (for example, reception of valid message), errors, DMA transfers, and so on are implemented. The following types of events can generate an interrupt:

- Reception of a new PSI5 message
- Reception of a new SMC message
- Unread SMC message overwritten in the SMC buffer area (PSI5_SFR)
- Unread PSI5 message overwritten in the PSI5 buffer area (PSI5_PMRH, PSI5_PMRL)
- Dedicated interrupts for selectable errors, mentioned below:
 - Bit C is set
 - Bit E and/or EM is set
 - Bit T is set
 - No PSI5 frame was received in a defined frame slot (Bit F is set)
- ECU-to-sensor communication-related interrupts
 - The application tries to load new data into the data preparation register before the register has signaled that it is ready for new data. (Note that the new command will be ignored in this case.)
 - The application loads new data into the data buffer register before the register has signaled that it is ready for new data. (Note that the new command will be ignored in this case.)
 - The application loads new data into the data shift register before the register has signaled that it is ready for new data. (Note that the new command will be ignored in this case.)
- DMA related interrupts (transfer finished/FIFO_FULL/underflow)
- Apart from this a single one sum interrupt OR-ing all the above events.

Various interrupt signals that represent the above events are:

- PSI5 new message arrival interrupt (**Status flags in PSI5_NDSR**) This is asserted when any of the programmed storage PSI5 locations (max 32) receive a new PSI5

message (error/error free). This interrupt also indicates that the PSI5 receive register (PMRRL/PMRRH) has been updated with a new PSI5 message.

- SMC message, ECU-to-sensor communication, and Time Stamp related interrupts (**All Status flags in PSI5_GISR**): This is asserted during following events:
 - When a new SMC message is received in its corresponding slot OR
 - When there is a CRC error in any SMC message OR
 - When an unread message in some corresponding SMC location (in PSI5_SFR) gets overwritten by a new SMC message OR
 - When DPR is ready for receiving a new data OR
 - When DBR is ready for receiving a new data OR
 - When DSR is ready for receiving a new data OR
 - When there is an attempt to write to DPR while it is NOT ready OR
 - When there is an attempt to write to DBR while it is NOT ready OR
 - When there is an attempt to write to DSR while it is NOT ready OR
 - When data time stamp register is updated with new time stamp value OR
 - When the sync time stamp read register is updated with a new time stamp.
- PSI5 Message Overwrite Interrupt (**Status Flags in PSI5_OWSR**): This is asserted when any of the programmed storage locations in the memory (max 32) have an unread PSI5 message that is overwritten by a new upcoming PSI5 message. Corresponds to OWSR.
- PSI5 Error interrupt (**Status Flags in PSI5_EISR**): This is asserted when any of the programmed storage locations in the memory (max32) have an error in the PSI5 message. It is used only for PSI5 messages.
- DMA Interrupt (**Status Flags in PSI5_DSR**):
 - When the DMA transfer corresponding to conf2, conf3 or conf4 finishes OR
 - When the DMA transfer corresponding to SMC message gets over OR
 - When there is an FIFO_FULL/Underflow in the PSI5 RAM register area OR
 - When there is an underflow in the SMC message area.
- Global Interrupt: Single interrupt ORing all above interrupt events.

53.4.5.1 Error conditions

- C: CRC/parity error on received data, evaluated after recalculating CRC on received data and comparing with received CRC or recalculating parity on received data and comparing with received parity.
- E: Electrical error during any Tbit period (excluding Messaging field M0M1) during frame reception (not applicable during Idle state); that is, if at least one bit is absent during its Tbit period.
- EM: Electrical error on M0M1 field.
- T: Timing error during frame reception violates the slot boundary. Here, all the corner cases and conditions are taken care of (for example, the isolation of correct slot to correct message if the message is too early or too late with respect to its expected position, the crossing of the slot boundary when one frame overlaps the other, and so on).
- F: No frame received during a particular slot time in case of sync mode.

53.4.6 DMA support

- The PSI5 module generates separate DMA transfer requests for PSI5 message/diagnostic status and SMC messages.
- A single DMA request is used for PSI5 messages and diagnostic data. The request is triggered based on the selected DMA configuration (either PSI5 message or diagnostic data or both). After choosing the configuration, data is available on respective DMA pop-up registers.
- DMA requests for PSI5 messages and diagnostic registers are based on the watermark level of message buffers (to be treated as FIFO for the DMA). PSI5 message buffer FIFOs get data from the PSI5 receiver in the order they are received.
- DMA requests for SMC messages are based on successful storage of each SMC frame to the message buffers. The SMC message buffers get data from SMC extraction logic in fixed round-robin arbitration sequence.
- Please see [Section 53.4.6.1, Single DMA request for PSI5/diagnostic message buffers](#) for PSI5 DMA handling and [Section 53.4.6.2, Single DMA request for SMC received messages](#) for SMC DMA handling

53.4.6.1 Single DMA request for PSI5/diagnostic message buffers

A common DMA request (per PSI5 block) is generated for PSI5 message/diagnostic buffers. The depth (up to 32 in present context) of the message buffer is programmable via the MEM_DEPTH filled in the PCCR. The width of each location of the PSI5 message buffer is 64 bits. When reading PSI5 message through the DMA, the 32-bit DPMR is sequenced with the data from the PSI5 message buffer. When reading the diagnostic registers through the DMA, the 32-bit DDSR is sequenced as NDSR followed by EISR. The DMA operation will happen as follows:

- Based on the DMA configuration chosen, one of the three DMA transfer sequence is initiated:
 - PSI5 buffer followed by diagnostic data sequenced to DPMR
 - PSI5 data sequenced to DPMR
 - Diagnostic buffer data sequenced to DDSR
- The DMA request is asserted when the number of messages stored in the buffer crosses the configured watermark number of messages.
- DMA request is deasserted once the “watermark” number of messages have been read.
- If the number of unread messages is still greater than the watermark programmed, then the DMA request is reasserted.
- The received message data is read from a common memory-mapped 32-bit DMA PSI5 message register (DPMR). Thus, a single PSI5 message (64 bits wide) will require two IPS read operations. The diagnostic data is read either through the DPMR or through the DDSR (single 32-bit register) by the DMA controller via the IPS interface.
- It is assumed that the DMA Controller's Transfer Control Descriptor (TCD) would be set to transfer bytes equal to the messages indicated by watermark level, as described in $\text{DMA_PM_DS_WM}[4:0]$ in [Section 53.3.2.3, DMA Control Register \(PSI5_DCR\)](#). It is assumed that when responding to a DMA request, the DMA controller will read faster

than the rate at which messages are received and stored in the buffer/FIFO, which will cause the DMA request to deassert.

- The next received message will be updated on the common memory map register once the previous message is read by the DMA (by the DMA sequencer).
- The store signal from the synchronization logic will push the message into the memory-mapped RAM registers and in parallel also into the PSI5 receive register. [Figure 1186](#) shows the format in which the messages will be stored in any message buffer (RAM registers) and finally when being read from DMA access register.
- For more details about DMA programming please refer to [Section 53.3.2.2, PSI5 Channel Control Register \(PSI5_PCCR\)](#), [Section 53.3.2.3, DMA Control Register \(PSI5_DCR\)](#), [Section 53.3.2.4, DMA Status Register \(PSI5_DSR\)](#), [Section 53.3.2.10, DMA PSI5 Message Register \(PSI5_DPMR\)](#) and [Section 53.3.2.12, DMA Diagnostic Status Register \(PSI5_DDSCR\)](#).

53.4.6.2 Single DMA request for SMC received messages

A separate DMA request (each channel) is generated for the SMC message buffer. Each SMC received messages would be stored in the internal registers and read out by DMA controller via a common memory-mapped DMA SMC Frame Register (DSFR) (single 32-bit register for each message buffer). The data flow will happen as follows:

- A request is asserted when the system buffer successfully receives one complete SMC message, and remains asserted until all complete messages have been read. Data is read from the DMA PSI5 Message Register in the memory map.
- It is the job of an internal sequencing logic to provide the next complete received message. The internal sequencer works in a round-robin fashion checking for the slot-specific registers for each SMC message. Thus the sequence will be slot1— slot2— slot3—slot4—slot5—slot6—slot1 and so on. In whichever slot it encounters the first complete SMC message, the DMA request is asserted and the sequencer will wait for this data to be read before moving on to the next slot in its loop. If a particular slot message remains unread and a new SMC message for the same slot comes in, then the previous message would be overwritten and the OW bit would be set.
- The DMA controller should be programmed to read only one message in one DMA transfer. Otherwise, if it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur. User software should ensure there is no overflow.
- For more details about DMA programming please refer “PCCR-PSI5 Channel Control Register” [on page 2137](#), “DCR-DMA Control Register” [on page 2143](#), “DSR-DMA Status Register” [on page 2146](#) and “DSFR-DMA SMC Frame Register” [on page 2155](#)

53.4.7 Message read through interrupt generation

There is a separate interrupt request for received messages for each PSI5 channel. Each PSI5 channel can be configured individually to generate an interrupt on reception of messages. An interrupt is asserted when the corresponding PSI5 channel has received a complete message (PSI5 or SMC). Please see [Section 53.4.5, Interrupt handling](#) for more details about interrupts. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing the corresponding PSI5 channel that is enabled to give out messages via interrupts through the IPS Interface. In response to these interrupts, the CPU is expected to read interrupt status register for

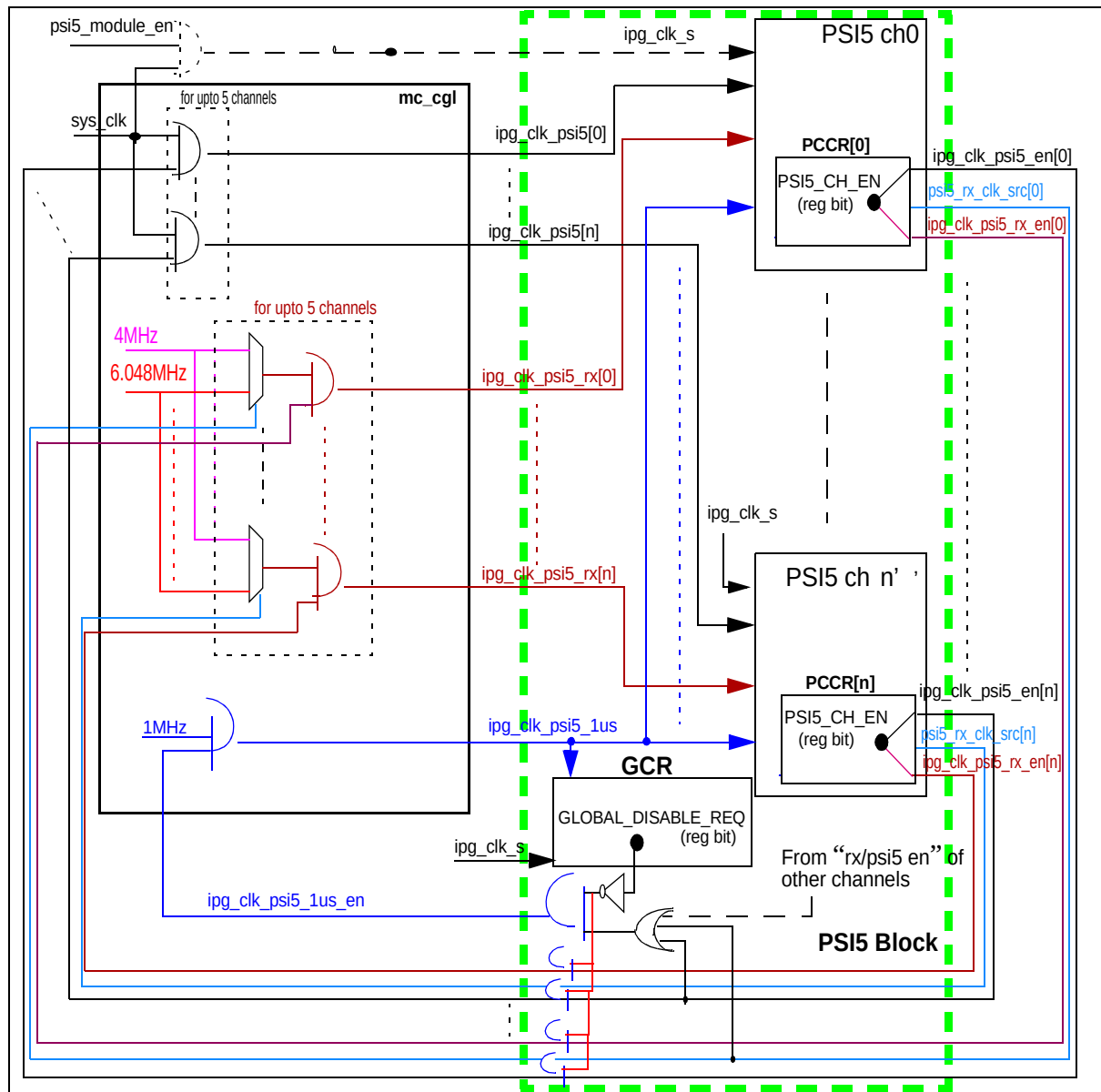
individual PSI5 Channel (depending on which interrupt was asserted) and store it in system memory.

53.5 Clock and Reset

The proposed clock scheme of the PSI5 block is shown in [Figure 1192](#). Global clock and clock divider are taken care of inside the CGM. Following is a brief description of these clocks with respect to the CGM:

- Global clock (please refer to [Figure 1192](#))
 - 1 Global clock generated from a single source but separately controllable for each PSI5 channel ("ipg_clk_psi5" for each channel that is generated from sys_clk).
 - It has a jitter of less than 5 ns at 1 MHz
 - 1 global clock, which is enabled during any register read/write from IPS interface. ("ipg_clk_s", which is common to all the PSI5 channels)
- Clock divider
 - Only one clock divider for all PSI5 interfaces (divider to generate 4 MHz/6.048 MHz from the system clock)
 - The clock divider generates all clocks that are needed for up to five interfaces
 - The clock divider is configurable by software
 - The output of the clock divider has a jitter of less than 5 ns at 1 MHz
- Separate GTM clock (not shown in [Figure 1192](#)) for each channel, which is available from at the system level. For reliable operation, the GTM clock should be limited in frequency to up to 3.9 MHz.

Figure 1192. PSI5 clock scheme



Note: Refer to the Device Configuration chapter for value of `PSI5_NUM_CH`.

53.5.1 Latency and Jitter related requirements

- Latency related
 - From the detection of the last bit of a frame (end of last CRC bit of message) at the input pin of the controller, up to provision of the data for the software (including CRC calculation): $\leq 3\mu\text{s}$.
 - From the sync pulse trigger output of the Sync Pulse Generator, up to the output of the sync pulse trigger at the data output pin of the controller: $\leq 2\mu\text{s}$.
- Jitter related
 - From the detection of the last bit of a frame (end of last CRC bit of message) at the input pin of the controller, up to provision of the data for the software (including CRC calculation): $\leq 1T_{\text{ipg_clk_psi5}}$
 - From the sync pulse trigger output of the Sync Pulse Generator, up to the output of the sync pulse trigger at the data output pin of the controller: $\leq 1\mu\text{s} + T_{\text{ipg_clk_psi5}}$

53.5.2 Reset

The device has a single global asynchronous reset input to the module that resets the whole module bringing it to an inactive state upon its assertion. The user software must program the module registers to bring up the receiver operations.

53.6 Initialization information

53.6.1 Initialization

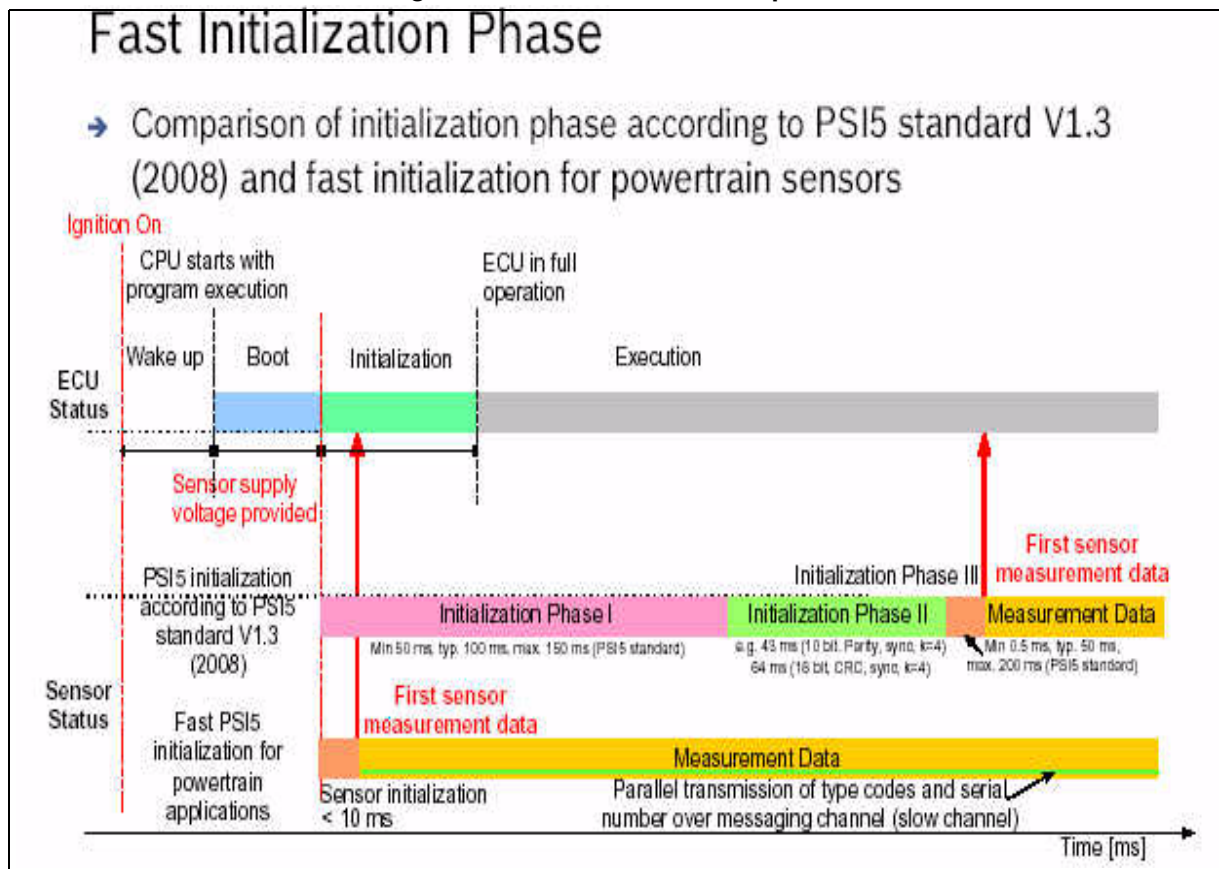
53.6.1.1 Sensor startup and initialization

After each power-on or under-voltage reset, the sensor performs an internal initialization, which is divided into three phases:

- Initialization Phase1: No data transmission.
- Initialization Phase 2: Transmission of type code and serial number of sensor.
- Initialization Phase 3: Transmission of “Sensor Ready” or “Sensor Defect” or other sensor specific data.

As this PSI5 implementation is targeted towards powertrain devices/sensors the standard initialization phase is modified and tuned for those applications. The comparison of the original scheme and enhanced schemes are shown in [Figure 1193](#). [Figure 1193](#) also describes that the initialization content is the same as defined in V1.3, and the only modification is that it is transmitted using the SMC channel when a fast startup is required. “Fast Initialization Phase” below means that the sending of first sensor data is fast, but the sending of sensor initialization data will actually be much slower.

Figure 1193. Fast initialization phase



53.7 Design Assumptions and Limitations

53.7.1 Assumptions

- Slot Configuration Register would always be programmed with valid number of bits for frame configuration.
- There will be no Timing error bits (T) in async mode. Also, in this mode the IP waits indefinitely for the detection of a valid start sequence, hence the F bit (NFR) is not set in Async mode. Rest all of the errors are available (E, EM, and C).

53.7.2 Limitations

As this is a PSI5 receiver, designed primarily to only receive the PSI5/SMC data and store as it is into the RAM buffer. So, it would not be responsible neither to recognize the content of the received messages nor do any decision making based on the content. Similarly, for Command it would not recognize the content of command and would transmit as it is writing into corresponding DSR and reception of PSI5 messages would not in any way depend on the content of command as far as PSI receiver is concerned. Its the responsibility of SW to sequence the issue of commands across the PSI5 channels and thereby recognize the content of received message and infer it.

53.8 Miscellaneous descriptions

53.8.1 Internal SYNC Pulse generation and coordination across PSI5 channels

Each PSI5 channel has the Channel Target Pulse Registers (CTPR) and Channel Initialize Pulse Register (CIPR). The CTPR defines the period of the internal sync pulse generation and the CIPR defines the initial offset time with respect to the sync pulse trigger. The sync pulse trigger can be generated from either the PSI5_GCR[CTC_GED] global bit or the PSI5_PCCR[CTC_ED] channel specific bit.

When staggering of the channels is required then the PSI5_GCR[CTC_GED] global bit is used as a common trigger for all the PSI5 channels in which PSI5_PCCR[CTC_GED_SEL] bit is set to 1. Setting this bit to 1 triggers the internal sync pulse generators of all the PSI5 channels and the trigger of each channel is then automatically staggered with respect to the other channel, depending on the value of the CIPR.

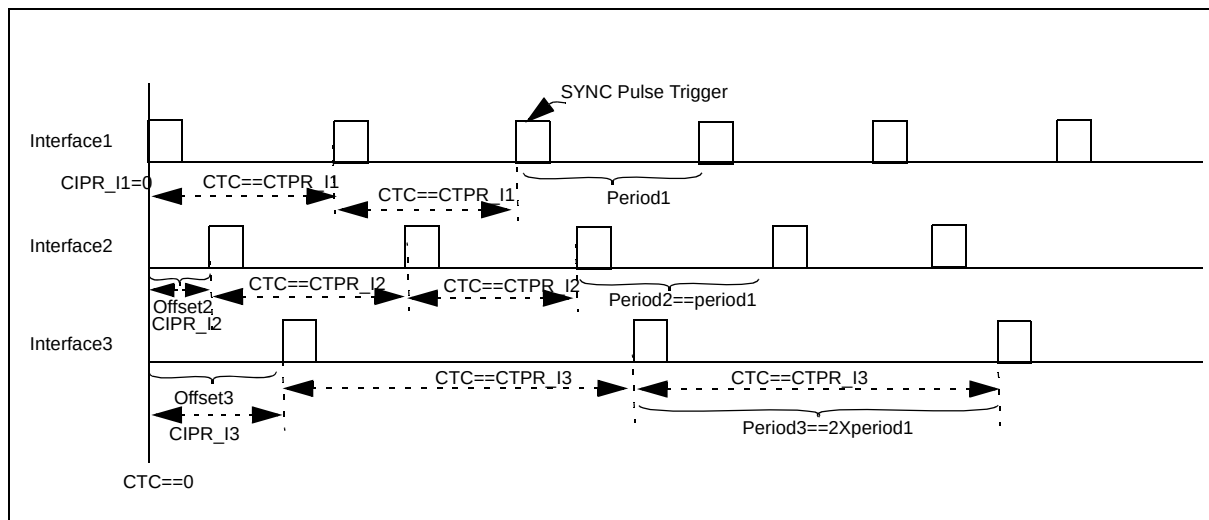
When the specific PSI5 channel is not to be kept as a part of staggered channels, then the channel-specific PSI5_PCCR[CTC_GED_SEL] bit has to be kept as 0. The channel in this case, has to be triggered locally by writing a 1 to the PSI5_PCCR[CTC_ED] of that specific channel. This triggers the internal Sync Pulse Generator of that specific channel without affecting other channels.

Following is the procedure for starting the internal Sync Pulse Generators:

- If in Normal mode, then stop and reset all Channel Trigger Counters (CTCs) by setting the global PSI5_GCR[CTC_GED] bit to 0 or the channel specific PSI5_PCCR[CTC_ED] to 0.
- Enter Config mode.
- In Config mode, load all CTCs with the initialization value (offset) from CIPR.
- In Config mode, load all Counter Target Pulse Registers (CTPR) with the needed sync pulse period.
- Enter Normal mode.
- Send global start trigger (using PSI5_GCR[CTC_GED] bit) to all CTCs for synchronized start for the staggered channels OR send a local start trigger (by using the PSI5_PCCR[CTC_ED]) for channels not staggered.
- If the Compare Unit detects that the value in the CTC is equal to the value in CTPR, the CTC is automatically reset by hardware and restarts with 0.
- In Normal mode, if required, the CTC can be stopped and reset by writing a "0" to the PSI5_GCR[CTC_GED] or the PSI5_PCCR[CTC_ED] as the case may be.
- A sync pulse trigger is created. (See below.)

For the details about the operation of the below interface, please see [Section 53.3.2.34, Counter Target Pulse Register \(PSI5_CTPR\)](#), [Section 53.3.2.35, Counter Initialize Pulse Register \(PSI5_CIPR\)](#) and [Section 53.4.3.4.5, Sync Pulse Trigger Generation](#).

Figure 1194. CIPR and CTPR



[Figure 1194](#) shows the CIPR and the CTPR definition w.r.t the staggering of the channels. [Figure 1195](#) is a more detailed representation of these parameters along with the associated jitter, across different channels when these channels are staggered. (The figure also shows some other parameters, such as the modulated pulse width parameters and the Manchester decoder disable parameters.)

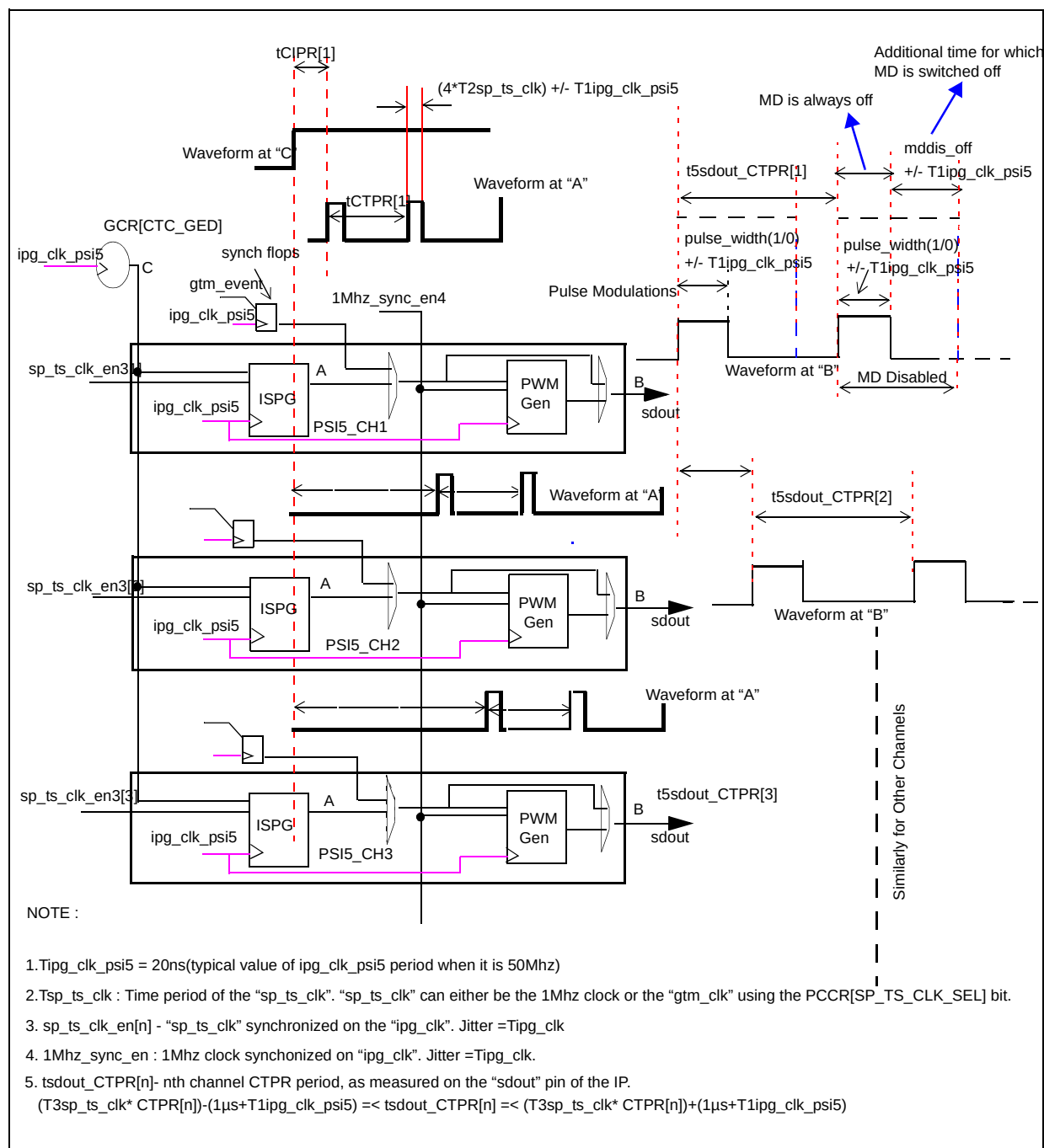
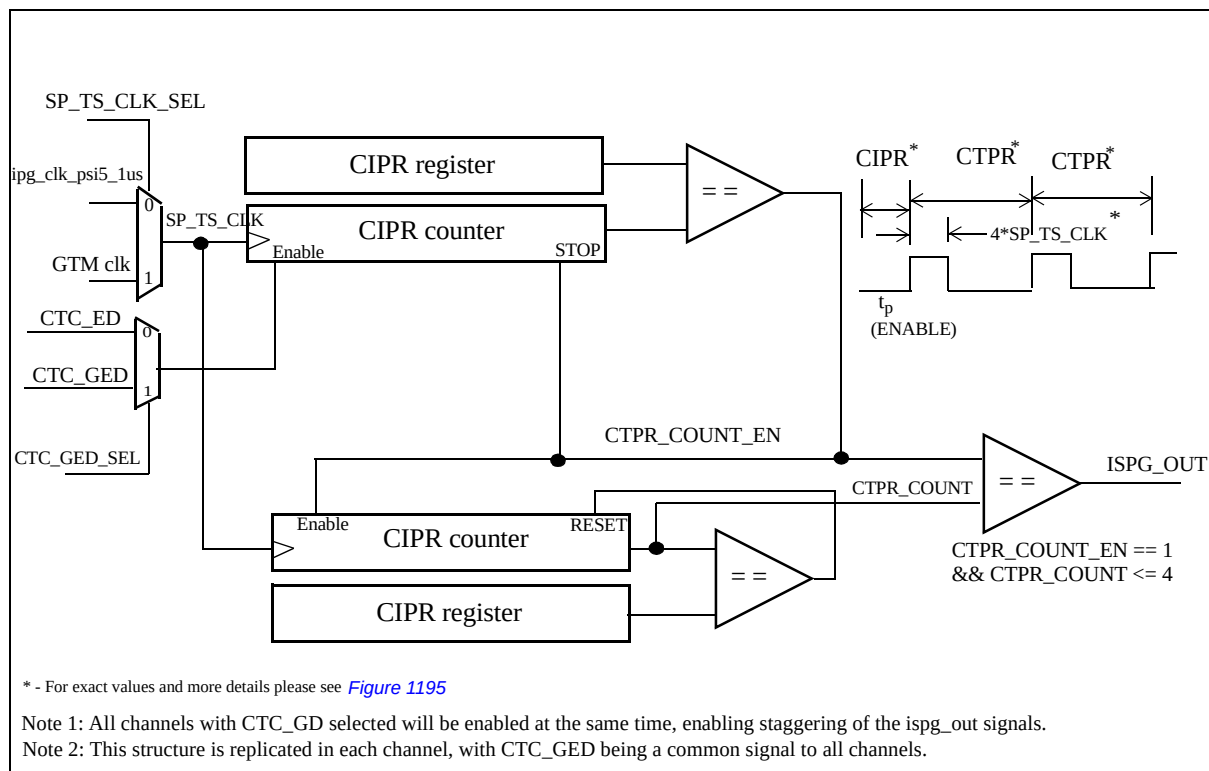


Figure 1195. Sync Pulse Generation Parameters and Jitters

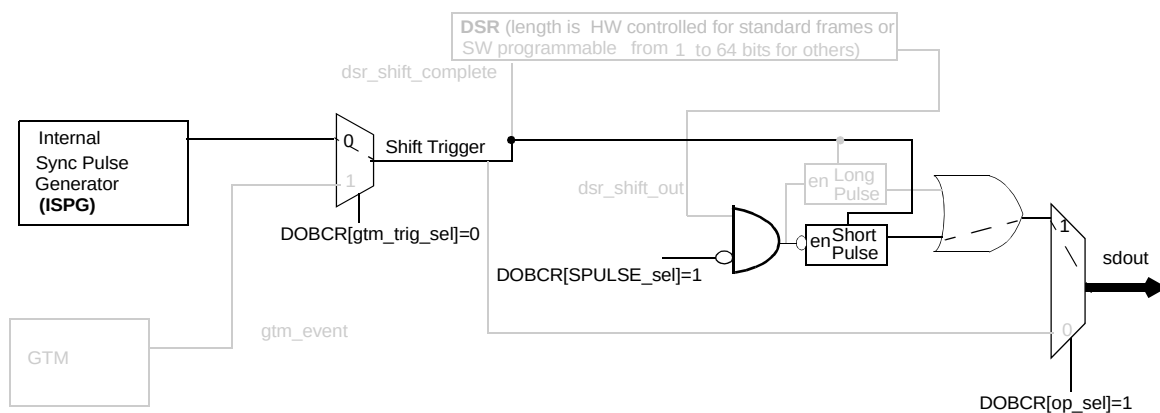
Figure 1195 shows the various jitters.

Figure 1196. ISPG(Internal Sync Pulse Generator) Structure

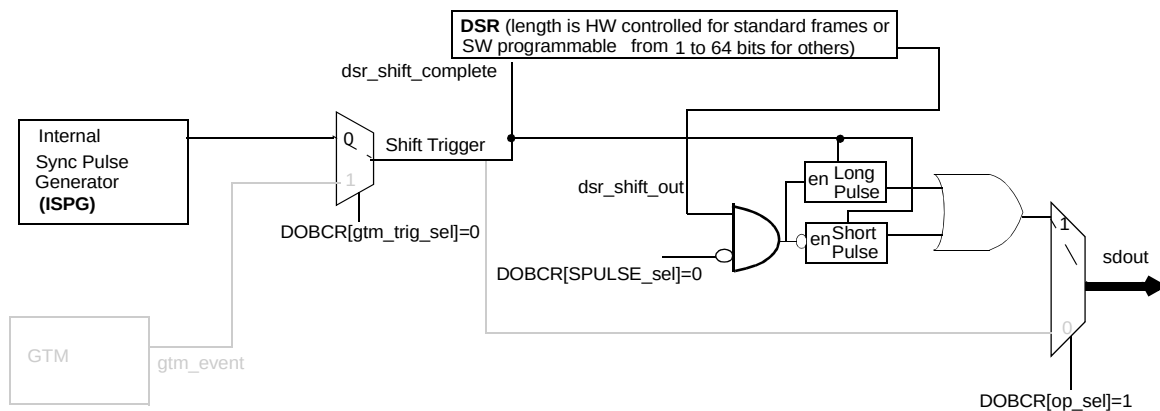


53.8.2 Valid states for integrated sync pulse generator

State1: Periodic Sync Pulse Generation (without pulse length modulation)



State2: Periodic Sync Pulse Generation with ECU-to-sensor Communication



State3: Event triggered (e.g. angle synchronous) sync pulses

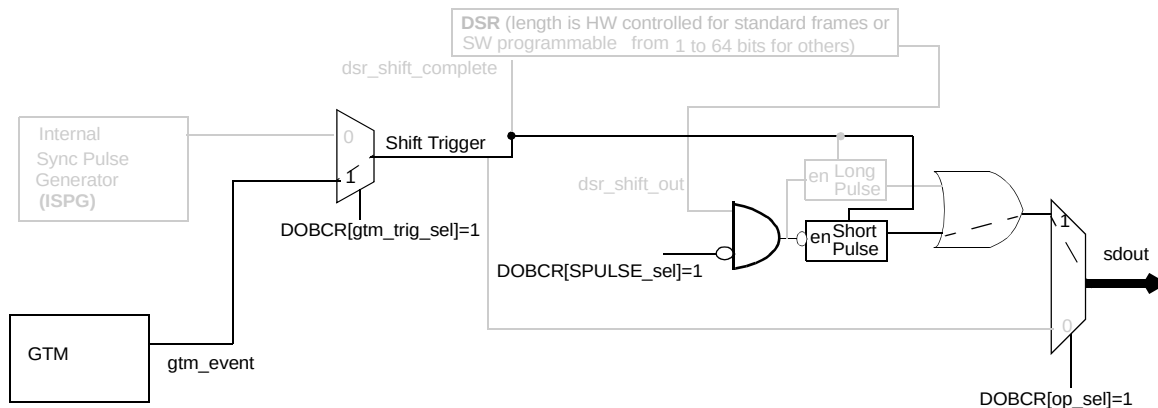
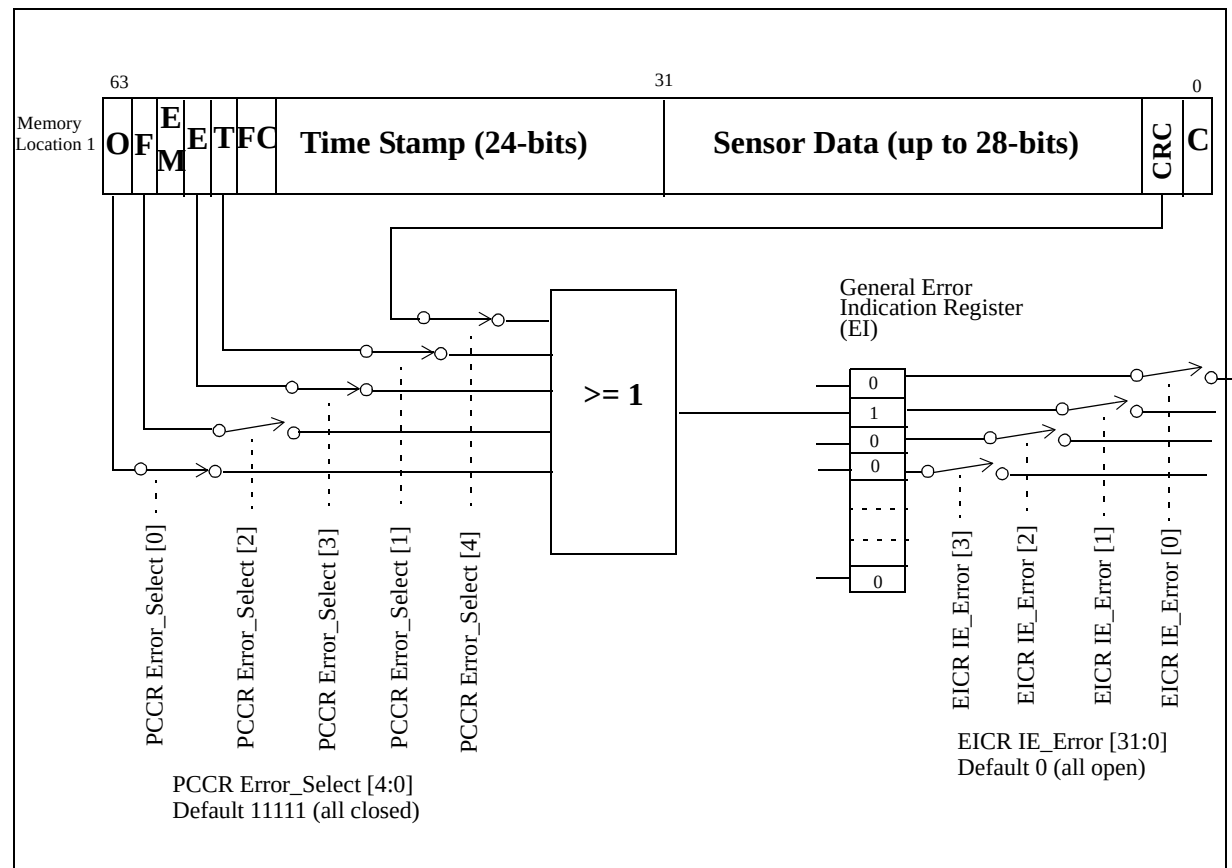


Figure 1197. ECU to Sensor Communication States

53.8.3 Implementation of EI Diagnostic Register

Figure 1198. Implementation of EI Diagnostic Register



54 Peripheral Sensor Interface-Support Module (PSI5-S)

54.1 Introduction

The PSI5 support IP (PSI5-S) module is used to interface directly to a UART compatible PSI5 transceiver over a high speed UART link. The transceiver is required to implement a compliant PSI5 module for all vehicle applications as is currently defined in the PSI5 protocol specification V2.0.

PSI5 messages from sensors received by the transceiver are decomposed into UART bytes and sent to the PSI5-S which in turn reconstructs the PSI5 messages from the UART packet , and makes them available for the application in system RAM.

ECU-to-sensor commands are sent from the PSI5-S module to the transceiver, which in turn generates the appropriate logical 1's and 0's in the SYNC pulses sent on the bus. The PSI5-S supports all command formats of the PSI5 V2.0 including the "tooth gap" method and "variable length" pulses.

At the system level, the PSI5-S supports one transceiver with up to seven PSI5 channels. Apart from these seven channels, there is an eighth channel available for special purposes, including direct communication with the transceiver and storage of corrupt messages.

54.1.1 Overview

PSI5 is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of safety airbag systems. The technical characteristics, the low implementation overhead, as well as the attractive cost make PSI5 also suitable for many other automotive sensor applications. The development goal of PSI5 is a flexible, reliable communication standard for automotive sensor applications that can be used and implemented license free.

The main features of PSI5 are high-speed and high-reliability data transfer at the lowest possible implementation overhead cost, offering a universal and flexible solution for multiple sensor applications. The main features of the PSI5 standard are:

- Two-wire current interface
- Manchester-coded digital data transmission from sensor to ECU
- High data transmission speed of 125 kbit/s or optional 189 kbit/s
- High EMC robustness and low emission
- Wide range of sensor supply current
- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication

The PSI5-S module can be configured to support the following PSI5 features. :

- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication

The PSI5-S module requires the physical and data link layer of the PSI5 to be implemented outside its interface. It also requires the PSI5 information to be embedded in the form of UART packet, from where it reconstructs the PSI5 message.

The module receives PSI5 message, from up to seven PSI5 channels each with upto six sequential frames. This is achieved via the external UART compliant transceiver. The PSI5 message in each of these frames are decomposed into three-to-six UART bytes(depending on PSI5 data payload) and streamed on a UART link at up to 6.25 Mbit/s to the PSI5-S module. In addition, via the PSI5-S interface, the application sends commands to the transceiver. The transceiver interprets them as either SYNC pulse commands (which will result in short, long or no SYNC pulses on the PSI5 bus), configuration commands (which will result in updates to the transceiver configuration) or diagnostic commands (which will result in the transceiver responding directly over the UART connection). In addition to the seven channels above there is an additional Channel, known as Channel0, which is used for special purposes like storing the direct responses from the transceiver or storing the unrecoverable messages.

The PSI5-S consists of three main functional blocks.

1. UART front end - Used as PSI5-S front end. When PSI5 functionality is not required then it is possible to use this as a standard UART (without LIN) also.
2. Message Recovery Unit - consisting of two state machines.
 - The first state machine interprets the input UART data stream and rebuilds the PSI5 message contained in the up to six UART bytes and also checks for the status of the correct reception of PSI5 message.
 - The second state machine sends SYNC pulse commands via the UART TxBuffer, to the external UART compliant transceiver, which in turn outputs SYNC pulses (short, long, or no SYNC pulse) on the PSI5 bus used to synchronize sensor responses in a bus configuration, and where required, also sends commands to sensors as defined in the PSI5 protocols V2.0.
3. PSI5 message interface - It is composed of two MRU buffers, MRU buffer1 and MRU buffer2. Both of them are collectively referred to as MRU output buffer. The MRU buffer1 is internal to the IP and not accessible from outside the IP. It is three word deep. MRU buffer2 is accessible to the application and is 4 word deep. MRU buffer2 can be read by the DMA interface, to transfer the stored PSI5 messages, to Channel specific mailboxes in system RAM directly.

Note:

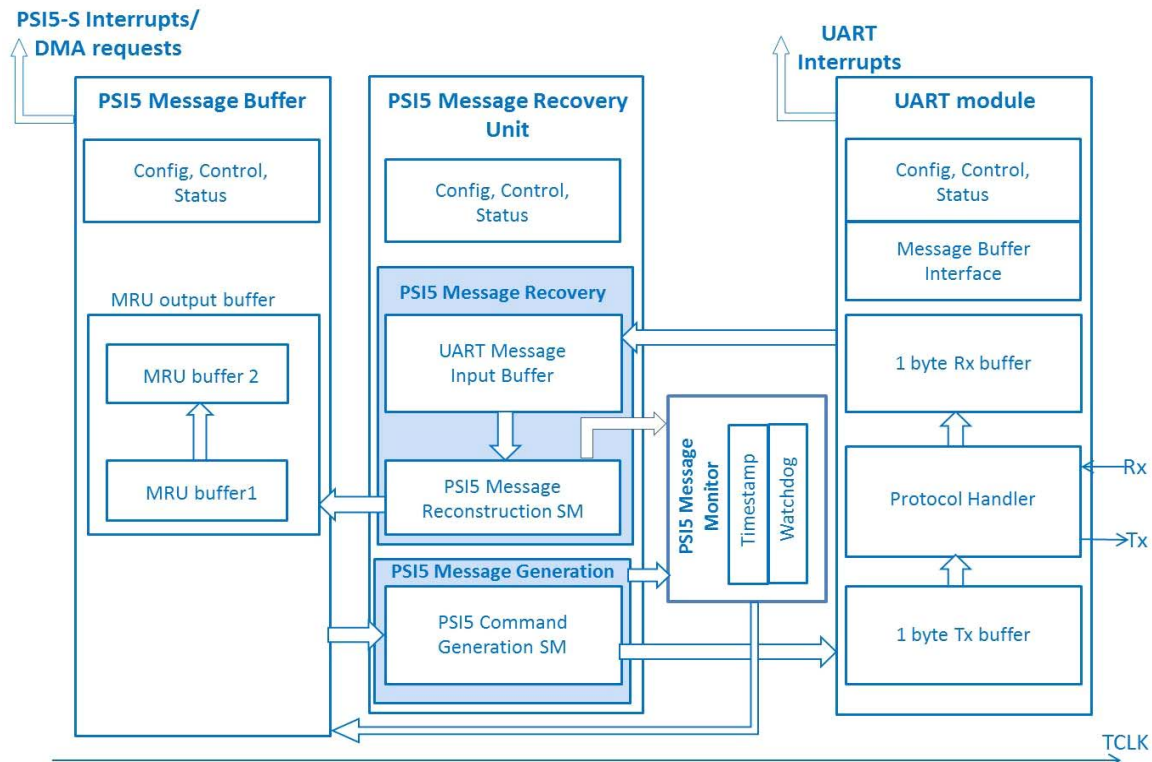
General Description

- *The location in system RAM where the messages are stored are referred to as the Mailbox . The mailbox is divided on the basis of channel and each channel is divided on the basis of slots on the PSI5 bus . Thus Mailbox1,location1 would mean Mailbox location for Channel1 and Slot No "1". This is also denoted by MBOX1,LOC1 or MBOX₁,LOC₁. This terminology would be followed in the subsequent sections of the doc.*
Each "location" of each mailbox channel is made up of 3x32 bits . The maximum size of each mailbox can be 6 "locations" . There are 8 such mailboxes . For channels operating in asynchronous mode , the whole of the Mailbox would act like a ring buffer and all the 6 locations would be filled in that manner.
Mailbox0 only has two locations MBOX0,LOC0 and MBOX0,LOC1. MBOX0,LOC0 is reserved for messages intended for Channel0(used for transceiver specific messages) while MBOX0,LOC1 contain "unrecoverable" messages for any channel . Other 4 locations of the MBOX0 are reserved.
- *Also a "UART byte" would mean the "byte" that is recieved by the UART reciever. "UART packet" would mean the group of UART bytes from which the PSI5 message would be extracted . "PSI5 message" means the message contained in the MRU*

buffers which contains the actual PSI5 message along with the timestamp and the error status bits.

Figure 1199 shows the block diagram of the PSI5-S module.

Figure 1199. PSI5-S digital interface block diagram



54.1.2 Features

- IPS interface for register read/write (big endian).
- Support for up to 7 PSI5 channels(Channel1 to Channel7) with each channel supporting up to 6 different Frame Streams (Slots). Each of the Frame Streams can be individually configured for Parity/CRC, payload Length, Synchronous/Asynchronous Operation, TimeStamp Capture event, Timestamp A or Timestamp B selection.
- Additional Channel0 for special functions.
- Supports bidirectional full duplex operations, via the UART Rx and Tx links.
- Provision of Extended CRC (XCRC) to check the coherence of the UART packet that comprise the PSI5 message.
- Message Reconstruction Unit (MRU) buffer2 (consisting of 4,32-bit words). Three of these words contain the PSI5 messages (PSI5 message payload, Message Status flags, Time Stamp) and the fourth one contains the mailbox address to which the three PSI5 words need to be transferred.
- A 24-bit time stamp register, clocked with either the module clock or the GTM clock(gtm_trig). The time stamp appended to the PSI5 message in the MRU, can be configured to be, either the one that is captured when a new PSI5 message is

- detected, or the one that is captured when the Output ECU to Sensor command is transferred from the UART TX buffer to the UART Tx Shift register.
- IPS interface DMA support, for transferring the data in the MRU output buffer to system memory.
 - Downstream Data Shift Register (DDSR) on per channel basis supporting the following :
 - Application can write the ECU to Sensor commands to the DDSR.
 - Supports automatic addition of the start bits/stuff bits /CRC calculation to the contents of DDSR.
 - Provision of resetting and rejecting the contents of DDSR at any stage of command transmission.
 - Support for conversion of the logic levels contained in the DDSR, to UART commands to be sent to the transceiver.
 - Provision of writing the Software calculated commands directly to the UART TX Buffer . This can be used for direct writing of the transceiver status and/or diagnostic commands.
 - Configurable UART message idle time counter, that indicates to the PSI5 MRU when an idle time at least as long as the configured length (number of bit times), has occurred on the Rx line.
 - Provision of configuring the PSI5-S front-end as a standalone UART module, when the PSI5-S module is disabled. In this mode the module, does not have any relation with the PSI5 protocol.
 - Support of up to 6.25 Mbit/s UART Rx Baud Rate.
 - An output clock for the PSI5 transceiver UART interface.

54.1.3 PSI5-S Global Modes of operation

The PSI5-S module can be configured in 4 different global modes of operation . These are UART_STDALONE,PS_DISABLE,PS_CONFIG and PS_NORMAL . These global modes can be entered by programming the PS_GLCR[GLOBAL_MODE] bits appropriately . [Figure 1200](#) shows the state transition diagram for these bits.

The PS_NORMAL,PS_CONFIG and PS_DISABLE are collectively referred to as the “PSI5-S” modes. Please see [Figure 1200](#). Once the PS_GLCR[GLOBAL_MODE] bits are appropriately configured for the “PSI5-S” modes , the information that the intended mode of operation is fully achieved is reflected by the setting of the PS_GLSR[GL_MODETR_DONE] status bit . An interrupt can also be generated for this event , in case the PS_GLCR[GL_MODETR_DONE_EN] bit is set to “1” . Thus, only when PS_GLSR[GL_MODETR_DONE] goes to “1” it should be considered that the IP has entered the desired “PSI5-S” mode.

This bit is reset to “0” when the mode is changed from any of the PSI5-S modes to the UART_STDALONE mode .

Following are the descriptions of the different global modes .

54.1.3.1 Standalone UART mode (UART_STDALONE mode)

This is the default state after the module reset. In this mode, only the PS_GLCR[GLOBAL_MODE] register bits and the registers of the UART module are accessible for writing. This is the mode which is to be used when the PSI5-S front end has

to be used as a standalone UART . This state indicates that the UART will now function as a standard UART and the PSI5-S core will be disabled. In this state, the PSI5-S register space is not accessible (except PS_GLCR[GLOBAL_MODE] registers) and any access to the registers of PSI5-S module (except PS_GLCR[GLOBAL_MODE] registers and UART specific registers) will generate transfer error.

54.1.3.2 Disable mode(PS_DISABLE)

PS_Disable mode in the PSI5 reduces power consumption. In this mode the RX, TX and common time base counters are disabled for any operation. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Please see [Table 1177](#) for the details of these bits. Note that when entering PS_Disable mode from any other mode, the register contents are retained and are not changed. The time stamp counter is however reset when entering Disable mode.

In “PS_Disable” mode, only PS_GLCR[GLOBAL_MODE] register bits are accessible for read/write. Accessing any other register bits will generate transfer error.

There is no software reset bit in the PSI5 module. The same has to be managed outside the IP at the system level. Channels will enter “Configuration/Normal mode” based on the value of the PS_GLCR[GLOBAL_MODE] bits.

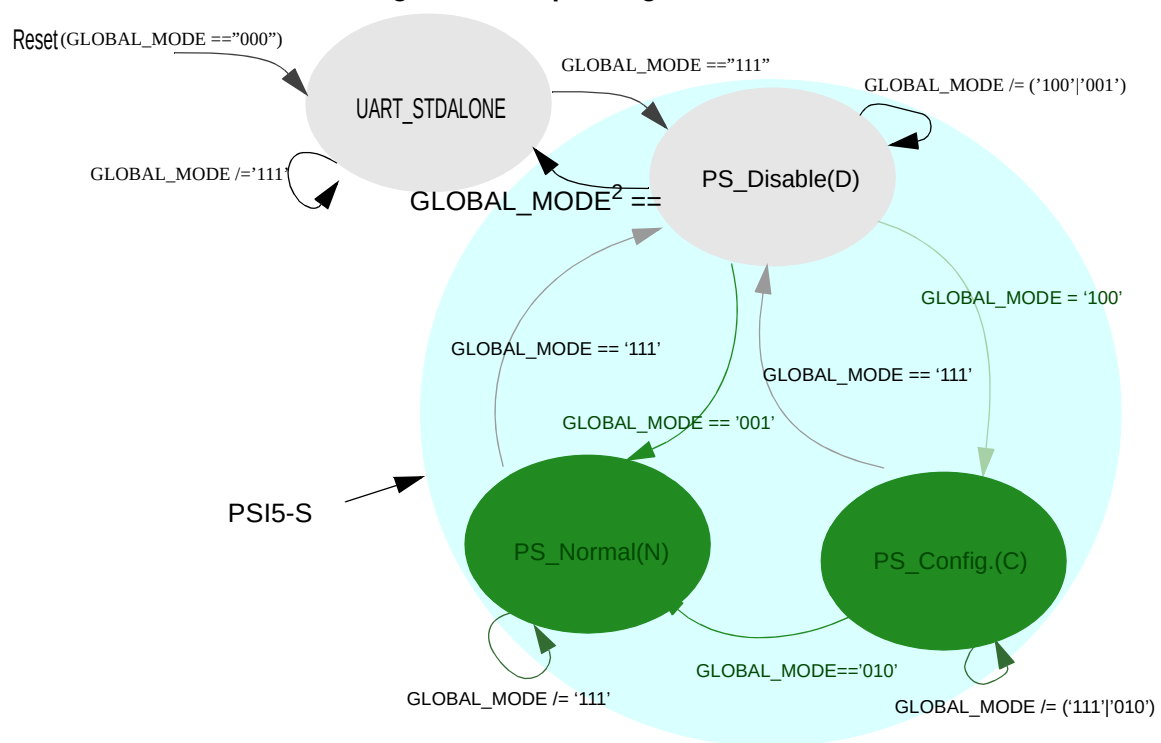
54.1.3.3 Configuration mode (PS_CONFIG mode)

In the configuration mode all PSI5 global, channel specific configuration/control registers and UART registers can be written. The receiver and transmitter sub-blocks remain disabled in this mode. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Please see [Table 1177](#) for the details of these bits.

54.1.3.4 Normal mode (PS_NORMAL mode)

In PS_NORMAL mode, the receiver and the transmitter blocks become active, error handling is enabled, UART starts its operation and all the PSI5 protocol functions are also enabled. In this mode, all the PSI5 global and channel-specific configuration registers are locked for writing, although these registers can still be read. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Please see [Table 1177](#) for the details of these bits. Note that there are still some register bits that can be written in this mode also. Please refer to the detailed register description.

Figure 1200. Operating modes



1. PSI5-S module can be configured in 4 different Global modes of operation . These are UART_STDALONE , PS_NORMAL,PS_CONFIG and PS_DISABLE . The PS_NORMAL,PS_CONFIG and PS_DISABLE are collectively referred to as "PSI5-S" modes.
2. PS_GLSR[GL_MODETR_DONE] status bit is reset to "0" when the transition occurs from PS_DISABLE to UART_STDALONE mode .

Table 1177. State transition Table

Present State	PS_GLCR (State Variables)			Next State
	PS_GLCR[GLOBAL_MODE(2)]	PS_GLCR[GLOBAL_MODE(1)]	PS_GLCR[GLOBAL_MODE(0)]	
UART_STDALONE	1	1	1	PS_Disable
PS_Disable	0	0	0	UART_STDALONE
PS_Disable	0	0	1	PS_Normal
PS_Disable	1	0	0	PS_Config
PS_Config	1	1	1	PS_Disable
PS_Config	0	1	0	PS_Normal
PS_Normal	1	1	1	PS_Disable

54.1.4 PSI5-S System Modes of operation

54.1.4.1 Stop mode

For PSI5-S modes:

In this mode , upon the STOP request from the system level the IP generates a system level acknowledgement with the information that the clocks to the IP can be switched off. The process of the same is as described below.

At the PSI5-S Rx side, as soon as stop mode request is recieved by the IP then the current PSI5 message under construction (in the internal PSI5 message extraction logic) is immediately saved to the MRU output buffer1 . Thus the assertion of the stop mode acts like a “forced” flushing for the current PSI5 message completion. Depending on the state of the PSI5 message under construction, the resultant PSI5 message might or might not have errors. Any byte that is being recieved by the UART is henceforth rejected. The internal PSI5 message extraction logic goes back to an “idle” state and remains so till the STOP mode request remains asserted.

At the PSI5-S Tx side , on the reception of the stop mode request , the internal sync pulse generation counters are stopped immediately(DDSR trigger counters) and no further command bits are shifted onto the UART Tx side . The UART Tx completes the current byte that is being transmitted and enters the idle state . No further transmissions take place.On deassertion of the stop mode , once the clocks restarts, the transmission of the commands resumes from where they had halted.

Once both the above processes are complete the IP generates acknowledgement for switching of the clocks to the PSI5-S IP.

For UART STDALONE mode:

For this please refer to the separate documentation describing the standalone LINFLEX module.

54.1.4.2 Debug mode

The IP enters this mode on system request. Any register can be read or written to by the debugger. The IP can enter the debug mode in addition to it being in any of the other modes (Uart Standalone/Disable/Normal/Config). In this mode the hardware clearing of the register PS_WDGTSSR remains disabled and any read of this register has no effect on the contents of this register. Apart from this there is no other difference as to whether the IP is in debug mode or not in the debug mode of operation.

54.2 External signal description

Table 1178. External signal description

S.No	Name	Function	I/O	Pull-up
Signals to/from pads				
1.	ipp_ind_linrx	LINUART receive pin. Used for Receiving UART bytes from transceiver which contain information to enable the rebuilding of the PSI5 message within the packet of between three and six UART bytes	I	Active
2.	ipp_do_lintx	LINUART transmit pin. Used for transmitting command bytes from TxFIFO to transceiver for generation of SYNC pulses (long, short or no) on the PSI5 bus or direct diagnostic/status commands for the transceiver itself	O	Passive
3.	ipp_port_en_tx	LINUART transmission port enable : Connected to the enable pin of of the pad which is to be used for "LINTX" functionality. When "1" it indicates that the IP is doing transmit of data.	O	Passive
4.	ipp_psi5_uart_tclk	Transceiver clock output, which drives the UART interface on the transceiver. The clock output frequency is configurable between 4 and 8 x UART baud rate on the interface between PSI5-S and transceiver.Frequency can be controlled at the system level through the Clock Generator Module(MC_CGM). The duty cycle of the transceiver clock is between 40% and 60%	O	Passive
IPS Interface				

54.3 Memory map and register description

54.3.1 Memory map

[Table 1179](#) shows the memory map.

Table 1179. PSI5-S memory map

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
0x0000	PSI5-S LIN Control register 1 (PS_LINCR1)	R	R/W	R	0x0000_0092	TBD	on page 2255
0x0004	PSI5-S LIN Interrupt Enable register (PS_LINIER)	R	R/W	R	0x0000_0000	TBD	on page 2257
0x0008	PS_LINSR(PS_NOUSE_REG ⁽³⁾)	Read value in PSI5-S modes is “dont care”. Also see ⁽⁴⁾ below.			0x0000_0040	—	on page 2258
0x000C	RESERVED	—	—	—	0x0000_0000	—	—
0x0010	PSI5-S UART Mode Control register (PS_UARTCR)	R	R/W	R	0x0000_0000	TBD	on page 2259
0x0014	PSI5-S UART Mode Status register (PS_UARTSR)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2262
0x00018	RESERVED	—	—	—	0x0000_0200	—	—
0x0001C	RESERVED	—	—	—	0x0000_FFFF	—	—
0x00020	RESERVED	—	—	—	0x0000_0E2C	—	—
0x0024	PSI5-S LIN Fractional Baud Rate register (PS_LINFBR)	R	R/W	R	0x0000_0000	TBD	on page 2264
0x0028	PSI5-S LIN Integer Baud Rate register (PS_LINIBRR)	R	R/W	R	0x0000_0000	TBD	on page 2265
0x0002C	RESERVED	—	—	—	0x0000_0000	—	—
0x0030	PSI5-S LIN Control register 2 (PS_LINCR2)	R	R/W	R	0x0000_0000	TBD	on page 2266
0x00034	RESERVED	—	—	—	0x0000_0000	—	—
0x0038	PSI5-S Buffer Data Register Least Significant (PS_BDRL)	R	R/W	R	0x0000_0000	TBD	on page 2268

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
0x003C	PSI5-S Buffer Data Register Most Significant (PS_BDRM)	R	R/W	R	0x0000_0000	TBD	on page 2269
0x0040	RESERVED	—	—	—	0x0000_0000	—	—
0x0044	RESERVED	—	—	—	0x0000_0000	—	—
0x0048	RESERVED	—	—	—	0x0000_0000	—	—
0x004C	PSI5-S Global Control register (PS_GCR)	R	R/W	R	0x0000_0000	TBD	on page 2270
0x0050	PSI5-S UART Preset Timeout register (PS_UARTPTO)	R	R/W	R	0x00000FFF	TBD	on page 2271
0x0054	UPSI5-S ART Current Timeout register (PS_UARTCTO)	R	R	R	0x0000_0000	TBD	on page 2272
0x0058	DMATXE(PS_NOUSE_REG ⁽³⁾)	For PSI5-S modes always program with 0x0000_00000 in Config Mode. Also see ⁽⁴⁾ below.			0x0000_0000	—	on page 2273
0x005C	DMARXE(PS_NOUSE_REG ⁽³⁾)	For PSI5-S modes always program with 0x0000_00000 in Config Mode. Also see ⁽⁴⁾ below.			0x0000_0000	—	on page 2274
0x0060– 0x009C	Reserved for UART registers					—	
0x00A0– 0x00B0	Reserved for UART registers					—	
PSI5-S GLOBAL REGISTERS							
0x00B4	Global Control Register (PS_GLCR)	R/W (5)	R/W	R/W (5)	0x0000_0000	TBD	on page 2274
0x00B8	Global Status Register (PS_GLSR)	R/ W1C (5)	R/ W1C (5)	R/ W1C (5)	0x0000_0000	TBD	on page 2277
0x00BC	Channel_Base_Address (PS_CH_BASE_ADDR)	R	R/W	R	0x0000_0000	TBD	on page 2278
0x00C0	MRU OUTPUT BUFFER2 REGISTER0 (PS_MRU_BUF2_REG0)	R	R	R	0x0000_0000	TBD	on page 2279

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
0x00C4	MRU OUTPUT BUFFER2 REGISTER1 (PS_MRU_BUF2_REG1)	R	R	R	0x0000_0000	TBD	on page 2280
0x00C8	MRU OUTPUT BUFFER2 REGISTER2 (PS_MRU_BUF2_REG2)	R	R	R	0x0000_0000	TBD	on page 2282
0x00CC	MRU OUTPUT BUFFER2 REGISTER3 (PS_MRU_BUF2_REG3)	R	R	R	0x0000_0000	TBD	on page 2282
0x00D0	PSI5-S Clock Divider "0" for WatchDog and DownStream Data Shift Register Trigger Counters . Used for Channel1 and Channel 2 . (PS_WD_DDTRIG_CLKDIV0)	R	R/W	R	0x0000_0000	TBD	on page 2283
0x00D4	PSI5-S Clock Divider "1" for WatchDog and DownStream Data Shift Register Trigger Counters . Used for Channel3 and Channel 4. (PS_WD_DDTRIG_CLKDIV1)	R	R/W	R	0x0000_0000	TBD	on page 2284
0x00D8	PSI5-S Clock Divider "2" for WatchDog and DownStream Data Shift Register Trigger Counters . Used for Channel5 and Channel 6. (PS_WD_DDTRIG_CLKDIV2)	R	R/W	R	0x0000_0000	TBD	on page 2285
0x00DC	PSI5-S Clock Divider "3" for WatchDog and DownStream Data Shift Register Trigger Counters . Used for Channel7. (PS_WD_DDTRIG_CLKDIV3)	R	R/W	R	0x0000_0000	TBD	on page 2286
0x00E0	PSI5-S Mbox Status Irq (PS_MBOX_SR_IRQ)	R	R	R	0x0000_0000	TBD	on page 2287
0x00E4	PSI5-S Error Status Irq (PS_ERR_SR_IRQ)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2288
0x00E8	PSI5-S Mbox select Irq0 (PS_MBOX_SEL_IRQ0)	R	R/W	R/W	0x0000_0000	TBD	on page 2292
0x00EC	PSI5-S Error Select Irq0 (PS_ERR_SEL_IRQ0)	R	R/W	R/W	0x0000_0000	TBD	on page 2294
0x00F0	PSI5-S Mbox Select Irq1 (PS_MBOX_SEL_IRQ1)	R	R/W	R/W	0x0000_0000	TBD	on page 2292
0x00F4	PSI5-S Error Select Irq1 (PS_ERR_SEL_IRQ1)	R	R/W	R/W	0x0000_0000	TBD	on page 2294

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
0x00F8	PSI5-S Mbox Select Irq2 (PS_MBOX_SEL_IRQ2)	R	R/W	R/W	0x0000_0000	TBD	on page 2292
0x00FC	PSI5-S Error Select Irq2 (PS_ERR_SEL_IRQ2)	R	R/W	R/W	0x0000_0000	TBD	on page 2294
0x0100	PSI5-S Mbox Select Irq3 (PS_MBOX_SEL_IRQ3)	R	R/W	R/W	0x0000_0000	TBD	on page 2292
0x0104	PSI5-S Error Select Irq3 (PS_ERR_SEL_IRQ3)	R	R/W	R/W	0x0000_0000	TBD	on page 2294
0x0108	PSI5-S Mbox Select Irq4 (PS_MBOX_SEL_IRQ4)	R	R/W	R/W	0x0000_0000	TBD	on page 2292
0x010C	PSI5-S Error Enable Irq4 (PS_ERR_SEL_IRQ4)	R	R/W	R/W	0x0000_0000	TBD	on page 2294
0x0110	PSI5-S Mbox Select Irq5 (PS_MBOX_SEL_IRQ5)	R	R/W	R/W	0x0000_0000	TBD	on page 2292
0x0114	PSI5-S Error Select Irq5 (PS_ERR_SEL_IRQ5)	R	R/W	R/W	0x0000_0000	TBD	on page 2294
0x0118	PSI5-S Mbox Select Irq6 (PS_MBOX_SEL_IRQ6)	R	R/W	R/W	0x0000_0000	TBD	on page 2292
0x011C	PSI5-S Error Select Irq6 (PS_ERR_SEL_IRQ6)	R	R/W	R/W	0x0000_0000	TBD	on page 2294
0x0120	PSI5-S Mbox Select Irq7 (PS_MBOX_SEL_IRQ7)	R	R/W	R/W	0x0000_0000	TBD	on page 2292
0x0124	PSI5-S Error Select Irq7 (PS_ERR_SEL_IRQ7)	R	R/W	R/W	0x0000_0000	TBD	on page 2294
0x0128	PSI5-S Watchdog Frame Error Status and Watchdog Frame Error Timestamp capture register (PS_WDGTSSR)	R	R/HC (6)	R	0x0000_0000	TBD	on page 2296
0x012C– 0x0168	RESERVED					—	—
Channel0							
0x016C	PSI5-S channel0 message configuration register A (PS_MSGA_CH0)	R	R/W	R	0x0000_0051	TBD	on page 2297
0x0170	PSI5-S channel0 message configuration register B (PS_MSGB_CH0)	R	R/W	R	0x0000_0000	TBD	on page 2298

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
0x0174	RESERVED					—	—
0x0178	PSI5-S Mailbox status register channel0 (PS_MBOX_SR_CH0)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2299
0x017C– 0x018C	RESERVED					—	—
Channel1							
0x0190	PSI5-S channel1 message configuration register A (PS_MSGA_CH1)	R	R/W	R	0x0000_0000	TBD	on page 2301
0x0194	PSI5-S channel1 message configuration register B (PS_MSGB_CH1)	R	R/W	R	0x0000_0000	TBD	on page 2303
0x0198	RESERVED					—	—
0x019C	PSI5-S Mailbox status register channel1 (PS_MBOX_SR_CH1)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2305
0x01A0	PSI5-S channel1 watchdog configuration register(PS_WD_CFGR_CH1)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2310
0x01A4	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH1)	R	R/W	R	0x0000_0000	TBD	on page 2311
0x01A8	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH1)	R	R/W	R	0x0000_0000	TBD	on page 2312
0x01AC	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH1)	R	R/W	R/W (5)	0x0000_8000	TBD	on page 2313
0x01B0	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH1)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2316
0x01B4	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSCR_H_CH1)	R	R	R	0x0000_0000	TBD	on page 2317
0x01B8	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSCR_L_CH1)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2318
0x01BC– 0x01C8	RESERVED					—	—

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
Channel2							
0x01CC	PSI5-S channel1 message configuration register A (PS_MSGA_CH2)	R	R/W	R	0x0000_0000	TBD	on page 2301
0x01D0	PSI5-S channel1 message configuration register B (PS_MSGB_CH2)	R	R/W	R	0x0000_0000	TBD	on page 2303
0x01D4	RESERVED					—	—
0x01D8	PSI5-S Mailbox status register channel2 (PS_MBOX_SR_CH2)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2305
0x01DC	PSI5-S channel1 watchdog configuration register(PS_WD_CFGR_CH2)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2310
0x01E0	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH2)	R	R/W	R	0x0000_0000	TBD	on page 2311
0x01E4	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH2)	R	R/W	R	0x0000_0000	TBD	on page 2312
0x01E8	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH2)	R	R/W	R/W (5)	0x0000_8000	TBD	on page 2313
0x01EC	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH2)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2316
0x01F0	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSR_H_CH2)	R	R	R	0x0000_0000	TBD	on page 2317
0x01F4	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSR_L_CH2)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2318
0x01F8– 0x0204	RESERVED					—	—
Channel3							
0x0208	PSI5-S channel1 message configuration register A (PS_MSGA_CH3)	R	R/W	R	0x0000_0000	TBD	on page 2301
0x20C	PSI5-S channel1 message configuration register B (PS_MSGB_CH3)	R	R/W	R	0x0000_0000	TBD	on page 2303
0x210	RESERVED					—	—

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
0x0214	PSI5-S Mailbox status register channel3 (PS_MBOX_SR_CH3)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2305
0x0218	PSI5-S channel1 watchdog configuration register(PS_WD_CFGR_CH3)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2310
0x21C	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH3)	R	R/W	R	0x0000_0000	TBD	on page 2311
0x0220	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH3)	R	R/W	R	0x0000_0000	TBD	on page 2312
0x0224	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH3)	R	R/W	R/W (5)	0x0000_8000	TBD	on page 2313
0x0228	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH3)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2316
0x022C	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSCR_H_CH3)	R	R	R	0x0000_0000	TBD	on page 2317
0x0230	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSCR_L_CH3)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2318
0x0234– 0x0240	RESERVED					—	—
Channel4							
0x0244	PSI5-S channel1 message configuration register A (PS_MSGA_CH4)	R	R/W	R	0x0000_0000	TBD	on page 2301
0x0248	PSI5-S channel1 message configuration register B (PS_MSGB_CH4)	R	R/W	R	0x0000_0000	TBD	on page 2303
0x024C	RESERVED					—	—
0x0250	PSI5-S Mailbox status register channel4 (PS_MBOX_SR_CH4)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2305
0x0254	PSI5-S channel1 watchdog configuration register(PS_WD_CFGR_CH4)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2310
0x0258	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH4)	R	R/W	R	0x0000_0000	TBD	on page 2311

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
0x025C	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH4)	R	R/W	R	0x0000_0000	TBD	on page 2312
0x0260	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH4)	R	R/W	R/W ⁽⁵⁾	0x0000_8000	TBD	on page 2313
0x0264	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH4)	R	R/W1C	R/W1C	0x0000_0000	TBD	on page 2316
0x268	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSCR_H_CH4)	R	R	R	0x0000_0000	TBD	on page 2317
0x26C	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSCR_L_CH4)	R	R/W ⁽⁵⁾	R/W ⁽⁵⁾	0x0000_0000	TBD	on page 2318
0x0270–0x027C	RESERVED					—	—
Channel5							
0x0280	PSI5-S channel1 message configuration register A (PS_MSGA_CH5)	R	R/W	R	0x0000_0000	TBD	on page 2301
0x0284	PSI5-S channel1 message configuration register B (PS_MSGB_CH5)	R	R/W	R	0x0000_0000	TBD	on page 2303
0X0288	RESERVED					—	—
0x028C	PSI5-S Mailbox status register channel5 (PS_MBOX_SR_CH5)	R	R/W1C	R/W1C	0x0000_0000	TBD	on page 2305
0X0290	PSI5-S channel1 watchdog configuration register(PS_WD_CFGR_CH5)	R	R/W ⁽⁵⁾	R/W ⁽⁵⁾	0x0000_0000	TBD	on page 2310
0X0294	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH5)	R	R/W	R	0x0000_0000	TBD	on page 2311
0x0298	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH5)	R	R/W	R	0x0000_0000	TBD	on page 2312
0x029C	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH5)	R	R/W	R/W ⁽⁵⁾	0x0000_8000	TBD	on page 2313
0x2A0	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH5)	R	R/W1C	R/W1C	0x0000_0000	TBD	on page 2316

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
0x2A4	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSR_H_CH5)	R	R	R	0x0000_0000	TBD	on page 2317
0x2A8	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSR_L_CH5)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2318
0x02AC– 0x2B8	RESERVED					—	—
Channel6							
0x02BC	PSI5-S channel1 message configuration register A (PS_MSGA_CH6)	R	R/W	R	0x0000_0000	TBD	on page 2301
0x02C0	PSI5-S channel1 message configuration register B (PS_MSGB_CH6)	R	R/W	R	0x0000_0000	TBD	on page 2303
0x02C4	RESERVED					—	—
0x02C8	PSI5-S Mailbox status register channel6 (PS_MBOX_SR_CH6)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2305
0x02CC	PSI5-S channel1 watchdog configuration register(PS_WD_CFGR_CH6)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2310
0x02D0	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH6)	R	R/W	R	0x0000_0000	TBD	on page 2311
0x02D4	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH6)	R	R/W	R	0x0000_0000	TBD	on page 2312
0x2D8	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH6)	R	R/W	R/W (5)	0x0000_8000	TBD	on page 2313
0x2DC	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH6)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2316
0x2E0	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSR_H_CH6)	R	R	R	0x0000_0000	TBD	on page 2317
0x2E4	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSR_L_CH6)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2318
0x02E8– 0x02F4	RESERVED					—	—

Table 1179. PSI5-S memory map (continued)

Offset from PSI5_BASE	Register	Access ⁽¹⁾			Reset value ⁽²⁾	Register classification	Location
		D	C	N			
Channel7							
0x02F8	PSI5-S channel1 message configuration register A (PS_MSGA_CH7)	R	R/W	R	0x0000_0000	TBD	on page 2301
0x02FC	PSI5-S channel1 message configuration register B (PS_MSGB_CH7)	R	R/W	R	0x0000_0000	TBD	on page 2303
0x0300	RESERVED					—	—
0x0304	PSI5-S Mailbox status register channel1 (PS_MBOX_SR_CH7)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2305
0x0308	PSI5-S channel1 watchdog configuration register(PS_WD_CFGR_CH7)	R	R/W (5)	R/W (5)	0x0000_0000	TBD	on page 2310
0x030C	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH7)	R	R/W	R	0x0000_0000	TBD	on page 2311
0x0310	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH7)	R	R/W	R	0x0000_0000	TBD	on page 2312
0x0314	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH7)	R	R/W	R/W (5)	0x0000_8000	TBD	on page 2313
0x0318	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH7)	R	R/ W1C	R/ W1C	0x0000_0000	TBD	on page 2316
0x031C	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSR_H_CH7)	R	R	R	0x0000_0000	TBD	on page 2317
0x0320	PSI5-S channel1 ECU to Sensor Upstream Data Shift register (PS_DDSR_L_CH7)	R	R/W (5)	R/W (5)	0x0000_0000	TB	on page 2318
0x0324– 0x330	RESERVED					—	—

1. In this column, R/W = Read/Write, R = Read-only, W = Write-only, W1C = write-1-to-clear and W1S = "write-to-1-shot" wherein writing a 1 causes an effect on the register bit while writing a 0 has no effect; a read always returning a 0, D = Disable mode, C = Config mode, N = Normal mode. All 32-bit registers are word/half word/byte addressable for write. Read is always word addressable. Trying to access any register space of PSI5-S, other than those mentioned in the memory map i.e. trying to access any "RESERVED" register in the memory space, results in the generation of transfer error. Note that the generation of transfer error is confined to this scenario only.
2. Locations from 0x0000 to 0x005C are for UART registers.
3. PS_NOUSE_REG : Stands for register NOT REQUIRED in PSI5-S modes. It will be required in UART STANDLONE mode.

4. Register has significance in UART STANDLONE mode. This document would only mention the bit field names in the corresponding register description. For detailed bit description and the functioning of this register in UART modes(sleep,init and normal) belonging to UART STANDLONE mode, please refer to the separate documentation describing the standalone LINFLEX module.
5. Not all register bits are of this type. Please see the register bit description for more details.
6. Register Bits are automatically hardware cleared on read of any bit of this register.

54.3.2 UART register configuration

This section contains information of the UART registers that are specifically required for correct functioning of the IP in the “PSI5-S” modes.

In order to operate the PSI5-S module, the UART configuration registers need to be programmed with appropriate values. The values that are required for different register fields are defined in the following register descriptions . Some bits have to be programmed with definite values of “1” or “0” only .

In the following UART register description “PS_NOUSE_BIT” against a bit name and bit description, means that the specific register bit is present in the UART module but it is NOT REQUIRED in the PSI5-S modes of operation. For correct operation in the PSI5-S modes , the corresponding register bit might be required to be programmed as “1” or a “0” ; the same would be mentioned in the bit description .

Note that a bit with “PS_NOUSE_BIT” against its description would still be required in the UART STANDALONE mode . For detailed information of the UART register bit functioning in the UART STANDLONE mode, please refer to the separate document describing the standalone LINUART module. There are some other register bits which dont have a “PS_NOUSE_BIT” against them . These can be programmed as required. A bit mentioned as “Reserved” would mean that neither it is available in the UART STANDLONE mode nor in the PSI5-S modes.

54.3.2.1 PSI5-S LIN Control Register 1 (PS_LINCR1)

The PS_LINCR1 register contains control bits used to configure features of the PSI5-S UART Front end.

Figure 1201. PSI5-S LIN Control Register 1 (PS_LINCR1)

Address: 0000h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	AUTOWU	0	0	0	0	1	SLFM	LBKM	1	0	RBLM	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0

1.

Table 1180. PS_LINCR1 field descriptions

Field	Description
Reserved	Reserved Read returns 0
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
AUTOWU	PS_NOUSE_BIT Always program as “0” in PSI5-S modes.
Reserved	Reserved
Reserved	Reserved
SLFM	Self test mode 0 Self test mode disabled 1 Self test mode enabled To be used along with the LBKM bit . When this bit is set then in addition to the Loop Back mode; the “ipp_do_lintx” is driven to “1” .
LBKM	Loop Back mode 0 Loop Back Mode disabled 1 Loop Back mode enabled Programmed in PS_CONFIG mode. When set then then the “ipp_do_lintx” output is internally looped back and connected to “ipp_ind_linrx” pin .
Reserved	Reserved Read Only bit . Always read as “1”.
Reserved	Reserved
RBLM	PS_NOUSE_BIT Always program as 0 in PSI5-S modes.
SLEEP	PS_NOUSE_BIT Read Only Bit in PSI5-S modes. R/W in UART Standlone mode.
INIT	PS_NOUSE_BIT Read Only Bit in PSI5-S modes. R/W in UART Standlone mode.

54.3.2.2 PSI5-S LIN Interrupt enable register (PS_LINIER)

Figure 1202. PSI5-S LIN Interrupt enable register (PS_LINIER)

Address: 0004h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	0	0	0	0	0	FEIE	BOIE	0	WUIE	0	TOIE	DRIE	DTIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1181. PS_LINIER field descriptions

Field	Description
Reserved	Reserved Read returns 0
SZIE	Stuck at Zero Interrupt Enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core
OCIE	PS_NOUSE_BIT Always program as 0 in PSI5-S modes.
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated if this bit is set and the Stuck at Zero Flag (FEF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core
BOIE	Buffer Overrun Error Interrupt Enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core

Table 1181. PS_LINIER field descriptions (continued)

Field	Description
Reserved	Reserved. Always keep as 0.
WUIE	PS_NOUSE_BIT Always program as 0 in PSI5-S modes.
Reserved	Reserved
TOIE	Timeout Interrupt Enable 0 No interrupt 1 Interrupt generation enabled. An interrupt is generated if this bit is set and the Timeout Flag (TO) in PS_UARTSR is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core
DRIE	Data Reception complete Interrupt enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated if this bit is set and the Data Recieved Flag (DRF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core
DTIE	Data Transmitted Interrupt enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in PS_UARTSR is set. This interrupt can be used by the Application to check when the UART buffer is ready to accept the next byte. This is used by the application when it wants to write a transceiver command directly to the UART Tx buffer.
Reserved	Reserved Always read as "0".

54.3.2.3 PSI5-S LIN Status Register (PS_LINSR)

This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes

Figure 1203. LIN Status Register (LINSR)

Address: 0x0008H

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS3	LINS2	LINS1	LINS0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 1182. LINSR field description

Field	Description
	Reserved
UART State [3:0]	PS_NOUSE_BITS. Read Value is "dont care" in PSI5-S modes
	Reserved

54.3.2.4 PSI5-S UART Mode Control Register (PS_UARTCR)

Figure 1204. PSI5-S UART Mode Control Register (PS_UARTCR)

Address: 0010h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIS	CSP			OSR				ROSE	NEF			PCE_TX/DTU	SBUR[1:0]		WLS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TDFL/TFC				RDFL/RFC				RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE_RX
W																WLO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1183. PS_UARTCR field descriptions

Field	Description										
MIS	PS_NOUSE_BIT Always program as “1” in PSI5-S modes.										
CSP	Configurable Sample Point These bits will decide the sample point during reduced over sampling (ROSE=1). CSP can take the following range of values for a certain over sampling rate: <table border="1"> <thead> <tr> <th>OSR</th><th>CSP</th></tr> </thead> <tbody> <tr> <td>4</td><td>2, 3</td></tr> <tr> <td>5</td><td>2, 3, 4</td></tr> <tr> <td>6</td><td>3, 4, 5</td></tr> <tr> <td>8</td><td>NA</td></tr> </tbody> </table> For any values of CSP other than the ones mentioned in the above table, data sampled is erroneous.	OSR	CSP	4	2, 3	5	2, 3, 4	6	3, 4, 5	8	NA
OSR	CSP										
4	2, 3										
5	2, 3, 4										
6	3, 4, 5										
8	NA										
OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled (ROSE=1). For proper functionality, OSR can only take values 4, 5, 6 and 8. For any other values start bit of data is not detected by receiver and thus the reception will never start.										
ROSE	Reduced Over Sampling Enable 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate. The default value of this bit is 0.										
NEF	PS_NOUSE_BIT Always program as “000” in PSI5-S modes.										
PCE_Tx/DTU	In PSI5-S mode Parity Control Enable for Tx path(In UART STANDLONE mode used to disable UART timeout-Refer to LINUART standalone documentation for more details) 1: Parity generation for Tx path enabled. 0: Parity generation for Tx path disabled. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[] == 1. When programmed as “1” then a 1 bit additional parity bit is added to the transmitted 8 bits of UART data, so a total of 9 bits including parity are present between a start and a stop bit , in the transmitted data When programmed as “0” then total of 8bits data is present between a start and a stop bit in the transmitted data.										
SBUR	PS_NOUSE_BIT Always program as “00” in PSI5-S modes.										
WLS	PS_NOUSE_BIT Always program as “0” in PSI5-S modes.										
TDFL/TFC	PS_NOUSE_BIT Always program as “000” in PSI5-S modes.										

Table 1183. PS_UARTCR field descriptions (continued)

Field	Description
RDFL/RFC	PS_NOUSE_BIT Always program as “000” in PSI5-S modes.
RFBM	PS_NOUSE_BIT Always program as “0” in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
TFBM	PS_NOUSE_BIT Always program as “0” in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
WL1	PS_NOUSE_BIT Always program as “0” in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
PC1	Parity Control {PC1,PC0} 00 Parity is Even for both the Rx/Tx paths 01 Parity is Odd for both the Rx/Tx paths. 10 A logical 0 is taken as parity bit for both the Rx and the Tx paths 11 A logical 1 is taken as parity bit for both the Rx and the Tx paths. Can be programmed only in the PS_CONFIG mode only and when PS_UARTCR[UART] == 1.
RxEn	Receiver Enable 0 Receiver disabled 1 Receiver enabled This bit can be programmed only when PS_UARTCR[] == 1.
TxEn	Transmitter Enable 0 Transmitter disabled 1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed This bit can be programmed only when when PS_UARTCR[UART] == 1
PC0	Parity Control {PC1,PC0} 00 Parity is Even for both the Rx/Tx paths 01 Parity is Odd for both the Rx/Tx paths. 10 A logical 0 is taken as parity bit for both the Rx and the Tx paths 11 A logical 1 is taken as parity bit for both the Rx and the Tx paths. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1.
PCE_Rx	Recieve Parity check enable . 1 : Enable the parity checking on the Rx path 0: Disable the parity checking on the Rx path Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1. When programmed as “1” then a 1 bit additional parity bit is expected along with the recieved 8 bits of UART data, so a total of 9 bits including partity are expected between a start and a stop bit , in the recieved data. When programmed as “0” then total of 8bits data is expected between a start and a stop bit in the recieved data.

Table 1183. PS_UARTCR field descriptions (continued)

Field	Description
WL0	PS_NOUSE_BIT Always program as “1” in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1.
UART	PS_NOUSE_BIT Always program as “1” in PSI5-S modes. Can be programmed in PS_CONFIG mode only

54.3.2.5 PSI5-S UART Mode Status Register (PS_UARTSR)

Figure 1205. PSI5-S UART Mode Status Register (PS_UARTSR)

Address: 0014H

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE3	PE2	PE1	PE0	RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF	DTF	NF
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C		W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2.

Table 1184. PS_UARTSR field descriptions

Field	Description
Reserved	Reserved Read returns 0
SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated on the UART interrupt lines if the SZIE bit in PS_LINIER is set.
OCF	PS_NOUSE_BIT Read Only Bit.
PE[3:1]	PS_NOUSE_BIT Read Only Bit.

Table 1184. PS_UARTSR field descriptions (continued)

Field	Description
PE0	Parity Error flag for Data0 byte These bits indicate if there is a Parity Error in the recieved byte. No interrupt is generated if this error occurs. 0 No parity error 1 Parity error in the corresponding received byte This flag is automatically cleared by hardware. The software can poll this flag for debugging purposes.
RMB	PS_NOUSE_BIT Read Only Bit.
FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). FEF is automatically cleared by hardware. An interrupt is generated if this bit is set and the FEIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core
BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and the previous byte has not yet been read from the UART Rx buffer due to some fault in PSI5-S core reading logic. It is automatically cleared by hardware . An interrupt is generated if this bit is set and the BOIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.
RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
WUF	PS_NOUSE_BIT Read Only Bit.
RFNE	PS_NOUSE_BIT Read Only Bit.
TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO is automatically cleared by hardware. An interrupt is generated if this bit is set and the TOIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.
DRF	Data Reception Completed Flag DRF is set by hardware in UART buffer mode and indicates that one byte has been recieved and is ready for being read. DRF is automatically cleared by hardware . An interrupt is generated if this bit is set and the DRIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose . It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.

Table 1184. PS_UARTSR field descriptions (continued)

Field	Description
DTF	Data Transmission Completed Flag DTF is set by hardware in UART buffer mode and indicates that data transmission is completed. DTF is automatically cleared by hardware. When the SW desires to write the commands directly to the UART Tx buffer for transmission then it can monitor this bit for an indication of when the UART Tx buffer is ready to accept the next 8 bits of the data. An interrupt will be generated if the DTIE bit in PS_LINIER is set.
NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

54.3.2.6 PSI5-S LIN Fractional Baud Rate Register (PS_LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

Figure 1206. PSI5-S LIN Fractional Baud Rate Register (PS_LINFBRR)

Address: 0024h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FBR3 ¹	FBR2 ¹	FBR1 ¹	FBR0 ¹
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in PS_CONFIG mode.

Table 1185. PS_LINFBRR field descriptions

Field	Description												
Reserved	Reserved Read returns 0.												
FBR[3:0]	Fractional Baud rates. Please see ^(Note:) , which follows Section 54.3.2.7, LIN Integer Baud Rate Register (PS_LINIBRR) . <table border="1"> <thead> <tr> <th>LINFBRR[4:0]</th><th>Fraction (LDIV)</th></tr> </thead> <tbody> <tr> <td>0h</td><td>0</td></tr> <tr> <td>1h</td><td>1/16</td></tr> <tr> <td>...</td><td>...</td></tr> <tr> <td>Eh</td><td>14/16</td></tr> <tr> <td>Fh</td><td>15/16</td></tr> </tbody> </table>	LINFBRR[4:0]	Fraction (LDIV)	0h	0	1h	1/16	...	...	Eh	14/16	Fh	15/16
LINFBRR[4:0]	Fraction (LDIV)												
0h	0												
1h	1/16												
...	...												
Eh	14/16												
Fh	15/16												

54.3.2.7 LIN Integer Baud Rate Register (PS_LINIBRR)

This register consists of control bits that decide the baud rate along with the PS_LINFBRR. It can be programmed only in Initialization mode.

Figure 1207. PSI5-S LIN Integer Baud Rate Register (PS_LINIBRR)

Address: 0028h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	IBR19 ⁽¹⁾	IBR18 ⁽¹⁾	IBR17 ⁽¹⁾	IBR16 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IBR15 ⁽¹⁾	IBR14 ⁽¹⁾	IBR13 ⁽¹⁾	IBR12 ⁽¹⁾	IBR11 ⁽¹⁾	IBR10 ⁽¹⁾	IBR9 ⁽¹⁾	IBR8 ⁽¹⁾	IBR7 ⁽¹⁾	IBR6 ⁽¹⁾	IBR5 ⁽¹⁾	IBR4 ⁽¹⁾	IBR3 ⁽¹⁾	IBR2 ⁽¹⁾	IBR1 ⁽¹⁾	IBR0 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in PS_CONFIG mode.

Table 1186. PS_LINIBRR field descriptions

Field	Description												
Reserved	Reserved Read returns 0.												
IBR[19:0]	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. Please see (Note:) below. <table> <tr> <th>LINIBRR[19:0]</th><th>Mantissa (LDIV)</th></tr> <tr> <td>00000h</td><td>LIN clock disabled</td></tr> <tr> <td>00001h</td><td>1</td></tr> <tr> <td>...</td><td>...</td></tr> <tr> <td>FFFFEh</td><td>1048574</td></tr> <tr> <td>FFFFFh</td><td>1048575</td></tr> </table>	LINIBRR[19:0]	Mantissa (LDIV)	00000h	LIN clock disabled	00001h	1	...	...	FFFFEh	1048574	FFFFFh	1048575
LINIBRR[19:0]	Mantissa (LDIV)												
00000h	LIN clock disabled												
00001h	1												
...	...												
FFFFEh	1048574												
FFFFFh	1048575												

Note: *UART baud rate*

The UART baud rate is programmed in two registers: the LIN Integer Baud Rate register(PS_LINIBRR) and the LIN Fraction Baud Rate register(PS_LINFBRR). The baud rate registers can be programmed only during the PS_CONFIG mode.

The baud rate is calculated with the following formula for both receiver and transmitter:

$$Tx_{BR} = Rx_{BR} = ipg_baud_clk / (16 \times LDIV) \quad (\text{when } UARTCR[ROSE] == 0)$$

$$Tx_{BR} = Rx_{BR} = ipg_baud_clk / (UARTCR[OSR] \times LDIV) \quad (\text{when } UARTCR[ROSE] == 1)$$

LDIV is an unsigned fixed point number which comprises of the mantissa and the fraction . The mantissa is coded into 20 bits of LINIBRR, and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled($UARTCR[ROSE] == 1$) then PS_LINFBRR is not used . Hence LDIV contains only the integer part programmed in LINIBRR. Following are 2 examples for two different baud rate calculations.

Example 1:

With $UARTCR[ROSE] = 0$.

$LDIV = 1041.67$, $ipg_baud_clk = 80$ MHz

$LINIBRR = 1041d = 411$ h

$LINFBRR = 11d = B$ h

$$Tx_{BR} = Rx_{BR} = 80 \text{ Mhz} / (16 \times 1041,67) = 4.8 \text{ Kbit/s}$$

Example 2:

When $UARTCR[ROSE] = 1$:

$LDIV = 10$ d, $LIN_CLK = 80$ MHz

$LINIBRR = 10$ d, $UARTCR[OSR] = 4$

$$Tx_{BR} = Rx_{BR} = ipg_baud_clk / (UART[OSR] \times LDIV) = 80 \text{ MHz} / (4 \times 10) = 2 \text{ Mbit/s}$$

54.3.2.8 PSI5-S LIN Control Register 2 (PS_LINCR2)

This register includes control status bits related to buffer operations.

Figure 1208. PSI5-S LIN Control Register 2 (PS_LINCR2)

Address: 0030h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ABRQ ²	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2. Register bit can be read/set by software.

Table 1187. PS_LINCR2 field descriptions

Field	Description
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. Transmission is aborted at the end of the current bit.
Reserved	Reserved Read Only bit.
Reserved	Reserved Read returns 0.

54.3.2.9 PSI5-S Buffer Data Register Least Significant (PS_BDRL)

Figure 1209. PSI5-S Buffer Data Register Least Significant (PS_BDRL)

Address: 0038h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_TX[3]								DATA_TX[2]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA_TX[1]								DATA_TX[0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1188. PS_BDRL field descriptions

Field	Description
DATA_TX[3]	PS_NOUSE_BIT Writing has no effect in fully configured ⁽¹⁾ PSI5-S mode.
DATA_TX[2]	PS_NOUSE_BIT Writing has no effect in fully configured ⁽¹⁾ PSI5-S mode.
DATA_TX[1]	PS_NOUSE_BIT Writing has no effect in fully configured ⁽¹⁾ PSI5-S mode.
DATA_TX[0]	Data Byte for Tx . Transmit Data byte of the data field.

1. All UART control registers are programmed as mentioned in this document.

54.3.2.10 PSI5-S Buffer Data Register Most Significant (PS_BDRM)

Figure 1210. PSI5-S Buffer Data Register Most Significant (PS_BDRM)

Address: 003Ch

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_RX[3]								DATA_RX[2]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA_RX[1]								DATA_RX[0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1189. PS_BDRM field descriptions

Field	Description
DATA_RX[3]	PS_NOUSE_BIT Always read as "0" in fully configured ⁽¹⁾ PSI5-S mode.
DATA_RX[2]	PS_NOUSE_BIT Always read as "0" in fully configured ⁽¹⁾ PSI5-S mode.
DATA_RX[1]	PS_NOUSE_BIT Always read as "0" in fully configured ⁽¹⁾ PSI5-S mode.
DATA_RX[0]	Recieve data byte for Rx Recieve data byte of the data field.

1. All UART control registers are programmed as mentioned in this document.

54.3.2.11 PSI5-S Global Control Register (PS_GCR)

Figure 1211. PSI5-S Global Control Register (PS_GCR)

Address: 0x004C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDFBM ¹	RDFBM ¹	TDLIS ¹	RD LIS ¹	STOP	SR ²
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in PS_CONFIG mode.
2. Register bit can be written only by software in PS_CONFIG mode. Bit is always read 0 by software.

Table 1190. PS_GCR field descriptions

Field	Description
Reserved	Reserved Forced to 0 by hardware
TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB: 0 The first bit of transmitted data is LSB — in other words, the first bit received is mapped on LSB bit (of DATA_TX in PS_BDRL register) 1 The first bit of received data is MSB — in other words, first bit received is mapped on MSB bit (of DATA_TX in PS_BDRL register)
RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB: 0 The first bit of transmitted data is LSB — in other words, the first bit received is mapped on LSB bit (of DATA_RX in PS_BDRM register) 1 The first bit of received data is MSB — in other words, first bit received is mapped on MSB bit (of DATA_RX in PS_BDRM register)
TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) . 0 Transmitted data is not inverted 1 Transmitted data is inverted
RD LIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) . 0 Received data is not inverted 1 Received data is inverted

Table 1190. PS_GCR field descriptions (continued)

Field	Description
STOP	PS_NOUSE_BIT Always program as “0” in PSI5-S modes.
SR	Soft reset SR executes a soft reset of the UART controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). This bit is always read as 0.

54.3.2.12 PSI5-S UART Preset Timeout Register (PS_UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed to monitor the idle state of the reception line. This register can be written by software any time.

Figure 1212. PSI5-S UART Preset Timeout Register (PS_UARTPTO)

Address: 0x0050

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PTO11	PTO10	PTO9	PTO8	PTO7	PTO6	PTO5	PTO4	PTO3	PTO2	PTO1	PTO0
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

1.

Table 1191. PS_UARTPTO field descriptions

Field	Description
Reserved	Reserved Read returns 0.
PTO[11:0]	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. The register must be programmed to a value of “Idle Timeout required in terms of number of bits recieved ” at that particular baud rate. Refers also to register UARTCTO.

54.3.2.13 PSI5-S UART Current Timeout Register (PS_UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see [Section 54.3.2.12, PSI5-S UART Preset Timeout Register \(PS_UARTPTO\)](#)) to monitor the idle state of the reception line.

The timeout counter:

- Starts at zero and counts upward
- Is clocked with baud clock frequency of T_{xBR}/R_{xBR} . Also refer ^(Note:) which follows [Section 54.3.2.7, LIN Integer Baud Rate Register \(PS_LINIBRR\)](#).
- Is automatically enabled when UARTCR[RXEN] = 1

Note: Due to internal synchronization there is a delay between 4 to 6 clock cycles of "ipg_clk_ps", for the internal counter's value (which is clocked with baud frequency of T_{xBR}/R_{xBR}) to reflect on to UARTCTO.

UARTCTO is reset when :

- UARTCTO becomes equal to UARTPTO
- Whenever UARTPTO is written.
- When the PSI5-S goes in PS_DISABLE mode.

Figure 1213. PSI5-S UART Current Timeout Register (PS_UARTCTO)

Address: 0x0054

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CTO11	CTO10	CTO9	CTO8	CTO7	CTO6	CTO5	CTO4	CTO3	CTO2	CTO1	CTO0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1192. PS_UARTCTO field descriptions

Field	Description
Reserved	Reserved Read returns 0.
CTO[11:0]	Current Timeout CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

54.3.2.14 DMA Tx Enable Register (DMATXE)

This register enables the DMA TX interface. This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes.

Figure 1214. DMA Tx Enable Register (DMATXE)

Address: 0x0058Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DTE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1.

Table 1193. DMATXE field descriptions

Field	Description
	Reserved Read returns 0.
DTE0	PS_NOUSE_BIT Always program with “0” in PSI5-S modes

54.3.2.15 DMA Rx Enable Register (DMARXE)

This register enables the DMA RX interface. This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes.

Figure 1215. DMA Rx Enable Register (DMARXE)

Address: 0x005C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1.

Table 1194. DMARXE field descriptions

Field	Description
	Reserved Read returns 0.
DRE0	PS_NOUSE_BIT Always program with "0" in PSI5-S modes

54.3.3 PSI5-S Register description

Unless specifically stated, the register bits are readable in all modes, but writable only in Config mode. Trying to write to any register outside its intended mode of write (Normal/config) will be ignored without the generation of any transfer error. PS_GLCR[GLOBAL_MODE] are the only register bits that can be written in the PS_DISABLE mode. In PS_DISABLE mode, accessing any other register bits will generate transfer error.

Note that, an access (read/write) to any register in the unimplemented register space will result in the generation of transfer error. All word locations are byte/half word/word accessible for write. Read is always a word access. Trying to write to "read only" bits of implemented register space, will be ignored without the generation of any error.

54.3.3.1 PSI5-S Global Control Register (PS_GLCR)

The Global Control Register (PS_GLCR) provides control functions for all the PSI5-S channels simultaneously.

Figure 1216. PSI5-S Global Control Register (PS_GLCR)

Address: 0x00B4

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GL_DDSR_TRIG	GL_MODETR_DONE_EN	MRU_ERR_EN	TSCS_B	TSCS_A	CLR_CNTR_B	CLR_CNTR_A	CLRTSCNT_G	CLRTSCNT_G_L	TSCNT_EN_B	TSCNT_EN_A	TSCNTEN_G	TSCNTEN_G_L	GLOBAL_MODE[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1195. PS_GLCR bit field details

Field	Description
Reserved	Reserved
GL_DDSR_TRIG	This is the global shift trigger of all the DDSRs of each channel. When the corresponding PS_E2SCR_CH[n][GL_TRIG_SEL] == 1 , only then is this bit effective, else this bit has no effect. 1 The shift triggers of all the DDSR's start synchronously. 0 Shift triggers of all the DDSR registers are halted (default state). If bit is cleared during runtime, PS_DDTRIG_PERR and PS_DDTRIG_OFFR counters are reset. <i>Note: If bit is subsequently set after being cleared , then the counters will restart and the offsets in PS_DDTRIG_PERR and PS_DDTRIG_OFFR registers will be applied again.</i> This bit is also writable in PS_NORMAL mode
GL_MODETR_DONE_EN	Interrupt enable for PS_GLSR[GL_MODETR_DONE] bit. 1: Generates the interrupt when PS_GLSR[GL_MODETR_DONE] bit is set. 0 : No interrupt is generated on setting of the PS_GLSR[GL_MODETR_DONE] bit This bit is also writable in PS_NORMAL and PS_DISABLE modes.
MRU_ERR_EN	Enables the interrupt for MRU overwrite when the unread contents of MRU Buffer1 are overwritten by a new message 0 MRU overwrite error is not enabled 1 MRU overwrite error is enabled <i>Note: This bit is also writable in ps_normal mode</i>

Table 1195. PS_GLCR bit field details (continued)

Field	Description
TSCS_B	This bit controls the selection of clock input to timestamp counter B, either by the external periodic clock or by the GTM module 1 gtm_trig is selected as the clock. 0 External clock(ipg_clk_ps_ts) is selected
TSCS_A	This bit controls the selection of clock input to timestamp counter A, either by the external periodic clock or by the GTM module 1 gtm_trig is selected as the clock . 0 External clock(ipg_clk_ps_ts) is selected
CLR_CNTR_B	This bit will reset timestamp counter B when PS_GLCR[CLRTSCNT_G] is not set 1 Timestamp counter B is reset This bit is a one shot and is always read as 0. <i>Note: This bit is also writable in ps_normal mode</i>
CLR_CNTR_A	This bit will reset timestamp counter A when PS_GLCR[CLRTSCNT_G] is not set 1 Timestamp counter A is reset This bit is a one shot and is always read as 0. <i>Note: This bit is also writable in ps_normal mode</i>
CLRTSCNT_G	This bit controls the global clearing of the TimeStamp counters A and B when CLRTSCNT_G_L == 1. When CLRTSCNT_G_L == 0 then this bit has no effect. 1 Both time stamp counter are cleared at the same time. This bit is a one shot and is always read as 0. <i>Note: This bit is also writable in ps_normal mode</i>
CLRTSCNT_G_L	This bit control whether the timestamp counters counter A and B are cleared simultaneously or independently: 0 Timestamp counters A and B are cleared independently. When PS_GLCR[CLR_CNTR_A] == 1 timestamp counter A is cleared. When PS_GLCR[CLR_CNTR_A]== 1 timestamp counter B is cleared. 1 Both timestamp counter are cleared simultaneously as soon as CLRTSCNT_G== 1.
TSCNT_EN_B	This bit controls enabling of timestamp counter A when PS_GLCR[TSCNTEN_G] is not set 0 Timestamp counter A not enabled 1 Timestamp counter B is enabled Note: This bit is also writable in ps_normal mode
TSCNT_EN_A	This bit controls enabling of timestamp counter A when PS_GLCR[TSCNTEN_G] is not set 0 Timestamp counter A not enabled 1 Timestamp counter A is enabled <i>Note: This bit is also writable in ps_normal mode</i>
TSCNTEN_G	This bit controls the global trigger of the TimeStamp counters A and B when TSCNTEN_G_L == 1. When TSCNTEN_G_ == 0 then this bit has no effect. 0 Both time stamp counters are disabled simultaneously 1 Both time stamp counter start simultaneously. <i>Note: This bit is also writable in ps_normal mode</i>

Table 1195. PS_GLCR bit field details (continued)

Field	Description
TSCNTEN_G_L	This bit control whether the timestamp counters counter A and B are started simultaneously or independently 0 Timestamp counters A and B are started independently. When PS_GLCR[TSCNT_A_EN] == 1 timestamp counter A starts. When PS_GLCR[TSCNT_B_EN] == 1 timestamp counter B starts. 1 Both timestamp counter are triggered simultaneously as soon as TSCNTEN_G == 1.
GLOBAL_MODE	These are the bits which decide Global Mode of the IP. There are 4 modes, UART_STDALONE, PS_DISABLE, PS_CONFIG and PS_NORMAL. For the details of modes related to the bit setting of GLOBAL_MODE, please refer Table 1177 . <i>Note: This bits are writable in all modes</i>

54.3.3.2 PSI5-S Global Status Register (PS_GLSR)

Figure 1217. PSI5-S Global Status Register (PS_GLSR)

Address: 0x00B8

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	GL_MODETR_DONE	MRU_ERR	0	0	0	0	0	0	0	FID[2:0]			CHID[2:0]		
W		W1C	W1C													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1196. PS_GLSR bit field details

Field	Description
Reserved	Reserved
GL_MODETR_DONE	This bit indicates , as to when the global mode transitions , as programmed by PS_GLCR[GLOBAL_MODE] have actually been accomplished : 1 The global mode as programmed in the PS_GLCR[GLOBAL_MODE] have been accomplished 0 The intended global mode , as programmed in the PS_GLCR[GLOBAL_MODE] register bits , is under processing. This bit is cleared by w1c. The bit is also writable in PS_NORMAL and PS_DISABLE mode. This bit is reset to “0” when the mode is changed from PS_DISABLE to the UART_STDALONE mode .
MRU_ERR	This bit shows whether unread MRU buffer1 contents have been overwritten by a new message: 1 overwrite occurred 0 no overwrite occurred This bit is cleared by w1c. The bit is also writable in PS_NORMAL mode.
Reserved	Reserved
FID[2:0]	Points to the frame whose data in MRU buffer 1 was overwritten
CHID[2:0]	Points to the channel whose data in MRU buffer 1 was overwritten

54.3.3.3 PSI5-S CHANNEL_BASE_ADDRESS (PS_CH_BASE_ADDR)

Figure 1218. PSI5-S Channel Base Address Register (PS_CH_BASE_ADDR)

Address:0x00BC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAILBOX_BASE_ADDR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MAILBOX_BASE_ADDR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1197. PS_CH_BASE_ADDR bit field details

Field	Description
MAILBOX_BASE_ADDR	This register will provide the base address of the mailbox in the system RAM.This value needs to be programmed by the software.

54.3.3.4 PSI5-S MRU OUTPUT BUFFER2 REGISTER0 (PS_MRU_BUF2_REG0)

Figure 1219. PSI5-S MRU Output Buffer2 Register0(PS_MRU_BUF2_REG0)

Address: 0x00C0

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHANNEL SPECIFIC MAILBOX ADDR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHANNEL SPECIFIC MAILBOX ADDR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1198. PS_MRU_BUF2_REG0 bit field details

Field	Description
CHANNEL SPECIFIC MAILBOX ADDR	This register will provide the channel specific mailbox address corresponding to channel and slot for which data has been received. This value is computed by the IP based on the PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR] and FID, CID. Following is the formula used to calculate this address: - For messages with CID==0 in the header byte, destined for Channel0 location0, the value is: $\text{PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR]}$ - For “unrecoverable” messages destined for Channel0, location1, the value is: $\text{PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR]} + 0xC$ - For other messages with CID = 1 to 7 the value can be calculated as: $(\text{PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR]} + (6 * 4 * 3 (\text{MSG_CH_ID}) + (\text{MSG_FRAME_ID}) * 4 * 3))$ When using the DMA for reading the MRU buffer, this register forms the Access1 of the DMA transfer.

54.3.3.5 PSI5-S MRU OUTPUT BUFFER2 REGISTER1 (PS_MRU_BUF2_REG1)

Figure 1220. PSI5-S MRU OUTPUT BUFFER2 REGISTER1 (PS_MRU_BUF2_REG1)

Address: 0x00C4

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DCI[3:0]				0	R_UVL_ERR	N_ERR	CHID[2:0]			FID[2:0]			R_OVL_ERR	F_WD_ERR	SCI_O_ERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SCI_F_ERR	SCI_P_ERR	HD_ERR	ERR[1:0]		CRC_ERR/ P_ERR	CRC[2:0]			XCRC_ERR	XCRC[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1199. PS_MRU_BUF2_REG1 bit field details

Field	Description
DCI[3:0]	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM
Reserved	Reserved
R_UVL_ERR	Message underflow. It is set when less than three UART bytes are received in the UART packet . In case of XCRC error the message is treated as unrecoverable and written in mail box location1 of channel 0 1 error occurred 0 no error occurred Please refer to Section 54.4.2.1, PSI5-S Message extraction for more details.
N_ERR	Indicates that the number of UART bytes in the recieved UART packet does not match the number specified in Channel Configuration Register A indicated by CID and FID i.e. the number of UART bytes does not match the value programmed in PS_MSGA_CH_CID[FID_bytes]. Also please see Fn_byte recalculation . 1 error occurred 0 no error occurred Please refer to Section 54.4.2.1, PSI5-S Message extraction for more details.
CHID[2:0]	Indicates Channel id of PSI5 message present in the UART header byte.
FID[2:0]	Indicates Frame ID of PSI5 message present in the UART header byte.

Table 1199. PS_MRUF2_BUF2_REG1 bit field details (continued)

Field	Description
R_OVL_ERR	Message overflow. It is set when more than six UART bytes are received in the UART packet. The message may be recoverable (XCRC_ERR == 0) or unrecoverable (XCRC_ERR == 1). Please refer to Section 54.4.2.1, PSI5-S Message extraction for more details. 1 error occurred 0 no error occurred
F_WD_ERR	Frame watchdog error in the received PSI5 message. This will be the F_WD_ERR of the specific channel indicated by the "CID", or it will be the logical OR of the F_WD_ERR of all the channels depending on the XCRC error status of the UART bytes. Please see Section 54.4.5, PSI5-S Watchdog Operation , Section 54.3.3.21, PSI5-S channel message configuration register B (PS_MSGB_CHn)[n=1..7] and Section 54.3.3.16, PSI5-S Watchdog Error Status and Watchdog Timestamp status register (PS_WDGTSSR) for more details. 1 error occurred 0 no error occurred
SCI_O_ERR	UART overrun error has occurred in at least one UART byte of the received UART packet, due to which some UART byte has been lost. 1 error occurred 0 no error occurred
SCI_F_ERR	UART framing error has occurred in the current received UART packet. 1 error occurred 0 no error occurred
SCI_P_ERR	UART message parity error has occurred in at least one UART byte of the received UART packet. 1 error occurred 0 no error occurred
HD_ERR	This bit is set when any of the ERR[1:0] is "1" and indicates a transceiver error. 1 error occurred 0 no error occurred
ERR[1:0]	Transceiver error flags received in the Message header byte of the UART packet indicated by CID and FID
CRC_ERR/ P_ERR	Indicates whether "PSI5 message" crc_err or the parity error in the PSI5 message indicated by CID and FID has occurred. The Parity is taken as even for PSI5 calculation: 1 indicates crc_error/Parity Error has occurred 0 no crc_error/parity error has occurred <i>Note: This bit always remains "0" for messages having CID and FID both as "0".</i>
CRC[2:0]	Contains the received CRC or Parity Bits of the PSI5 message, of indicated by CID and FID
XCRC_ERR	Indicates xcrc_err in the received UART packet: 1 indicates xcrc_error occurred 0 no xcrc_error occurred
XCRC[5:0]	This is the value of xcrc received in the last six bits of the UART byte, which occurs just before the information of a "new UART packet received" has been generated. Please refer to Section 54.4.1, PSI5-S UART Message Structure and Section 54.4.2.5, PSI5-S XCRC handling for more details.

54.3.3.6 PSI5-S MRU OUTPUT BUFFER2 REGISTER2 (PS_MRU_BUF2_REG2)

Figure 1221. PSI5-S MRU OUTPUT BUFFER REGISTER2 (PS_MRU_BUF2_REG2)

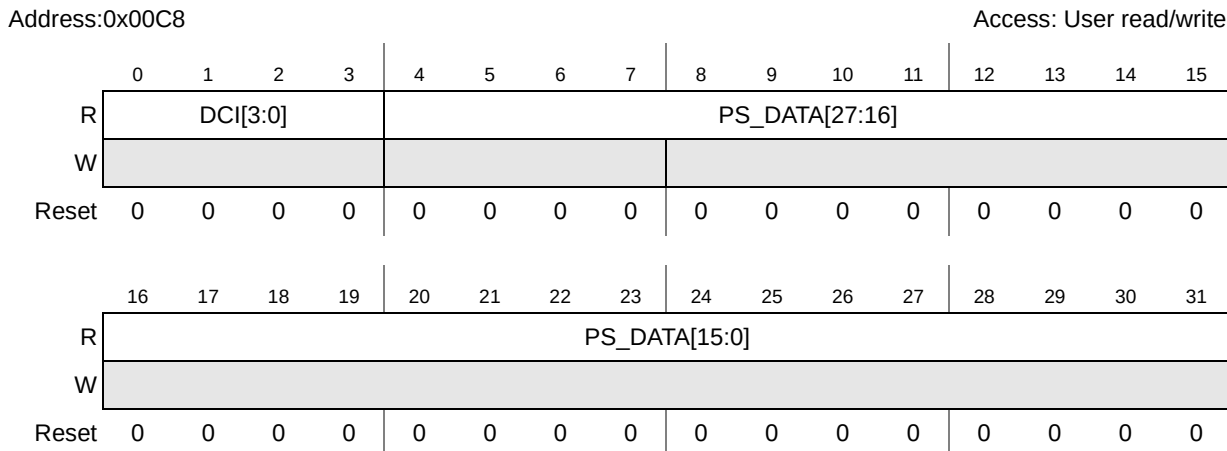


Table 1200. PS_MRU_BUF2_REG2 bit field details

Field	Description
DCI[3:0]	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM
PS_DATA[27:0]	PSI5 data bits extracted from the recieved UART bytes .

54.3.3.7 PSI5-S MRU OUTPUT BUFFER2 REGISTER3 (PS_MRU_BUF2_REG3)

Figure 1222. PSI5-S MRU OUTPUT BUFFER2 REGISTER3 (PS_MRU_BUF2_REG3)

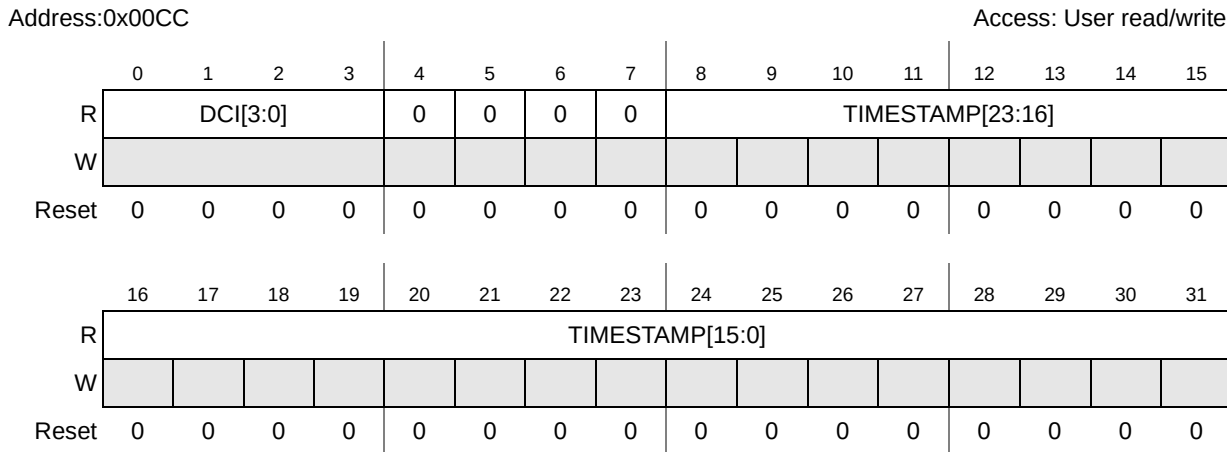


Table 1201. PS_MRU_BUF2_REG3 bit field details

Field	Description
DCI[3:0]	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM
Reserved	Reserved
TIMESTAMP[23:0]	Time stamp value appended based on the various timestamp bit settings in the PS_GLCR register, PS_PCCR_CH[n] register and the errors occurred. For more details please refer to Section 54.4.4, TimeStamp .

54.3.3.8 PSI5-S Clock Divider “0” for WatchDog and DownStream Data Shift Register Trigger Counters . Used for Channel1 and Channel2 (PS_WD_DDTRIG_CLKDIV0)

Figure 1223. PSI5-S Clock Divider “0” for WD and DDSR trigger counters (PS_WD_DDTRIG_CLKDIV0)

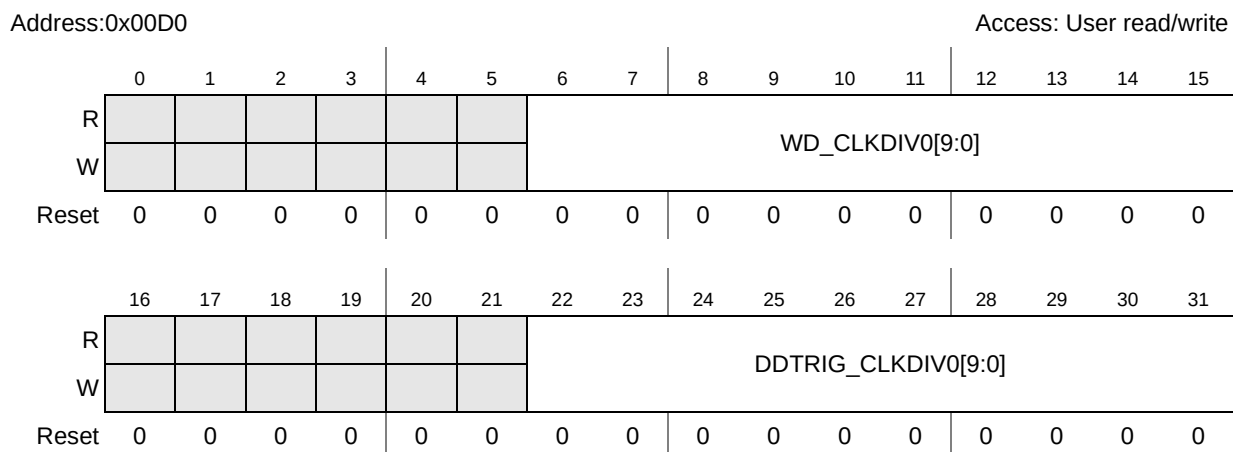


Table 1202. PS_WD_DDTRIG_CLKDIV0 bit field details

Field	Description
WD_CLKDIV0[9:0]	Clock Divider for watchdog clock . This register controls the division factor of the watchdog clock (ipg_clk_ps_wd, please see Figure 1258) in steps of 1us . This register controls the watchdog clock for Channel 1 and Channel2. Also please see Figure 1262 .
DDTRIG_CLKDIV0[9:0]	Clock Divider for DDSR shift trigger generation counters which in turn are used for sync pulse generation . This register controls the division factor of the DDSR trigger clock (ipg_clk_ps_ddtrig , please see Figure 1253) in steps of 1us . This register controls this clock for Channel 1 and Channel 2. Also please see Figure 1262 .

54.3.3.9 PSI5-S Clock Divider “1” for WatchDog and DownStream Data Shift Register Trigger Counters . Used for Channel3 and Channel4 (PS_WD_DDTRIG_CLKDIV1)

Figure 1224. PSI5-S Clock Divider “1” for WD and DDSR trigger counters (PS_WD_DDTRIG_CLKDIV1)

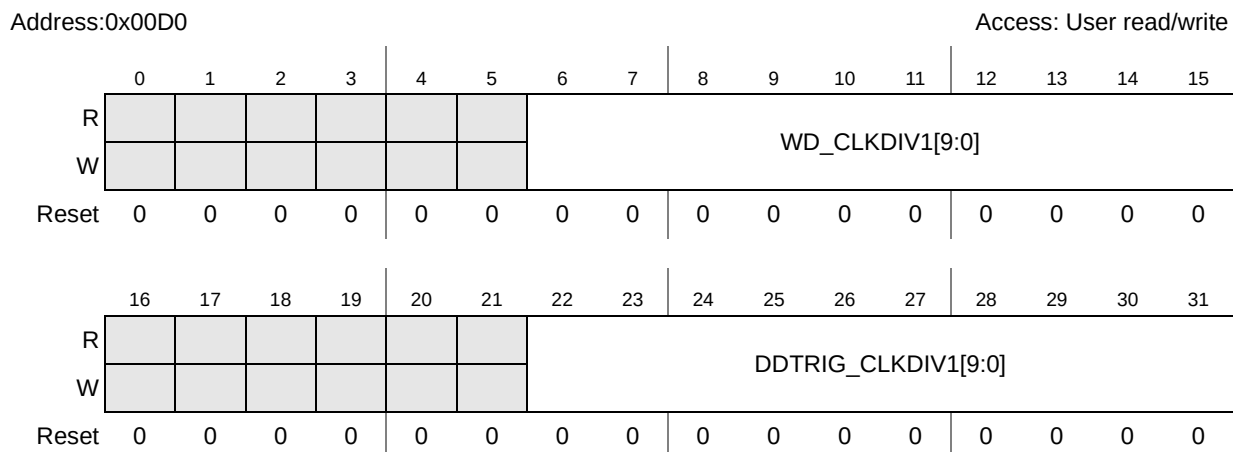


Table 1203. PS_WD_DDTRIG_CLKDIV1 bit field details

Field	Description
WD_CLKDIV1[9:0]	Clock Divider for watchdog clock . This register controls the division factor of the watchdog clock (ipg_clk_ps_wd, please see Figure 1258) in steps of 1us . This register controls the watchdog clock for Channel 3 and Channel 4. Also please see Figure 1262 .
DDTRIG_CLKDIV1[9:0]	Clock Divider for DDSR shift trigger generation counters which in turn are used for sync pulse generation . This register controls the division factor of the DDSR trigger clock(ipg_clk_ps_ddtrig , please see Figure 1253) in steps of 1us . This register controls this clock for Channel 3 and Channel 4. Also please see Figure 1262 .

54.3.3.10 PSI5-S Clock Divider “2” for WatchDog and DownStream Data Shift Register Trigger Counters . Used for Channel5 and Channel6 (PS_WD_DDTRIG_CLKDIV2)

Figure 1225. PSI5-S Clock Divider “2” for WD and DDSR trigger counters (PS_WD_DDTRIG_CLKDIV2)

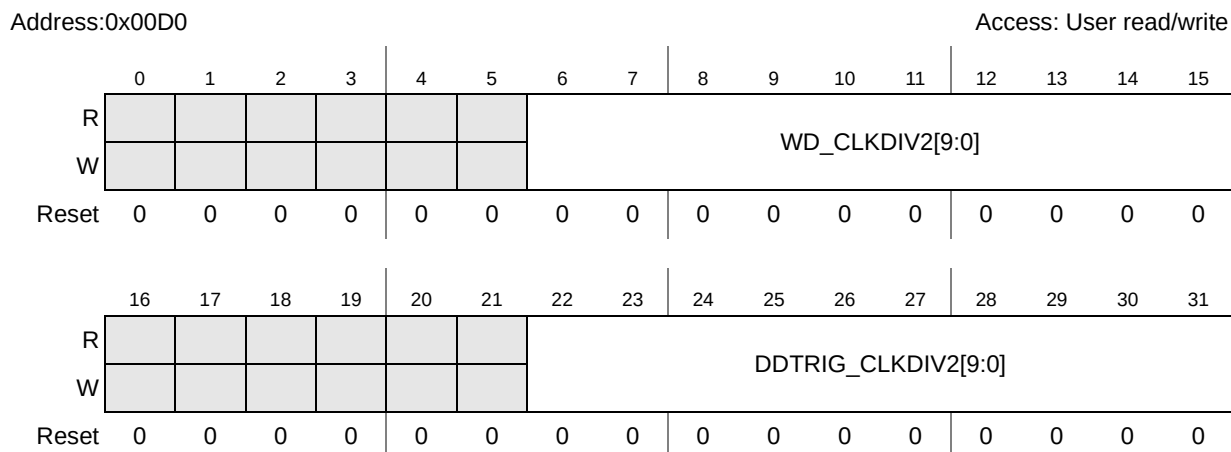


Table 1204. PS_WD_DDTRIG_CLKDIV2 bit field details

Field	Description
WD_CLKDIV2[9:0]	Clock Divider for watchdog clock . This register controls the division factor of the watchdog clock (ipg_clk_ps_wd, please see Figure 1258) in steps of 1us . This register controls the watchdog clock for Channel 5 and Channel 6. Also please see Figure 1262 .
DDTRIG_CLKDIV2[9:0]	Clock Divider for DDSR shift trigger generation counters which in turn are used for sync pulse generation . This register controls the division factor of the DDSR trigger clock(ipg_clk_ps_ddtrig , please see Figure 1253) in steps of 1us . This register controls this clock for Channel 5 and Channel 6. Also please see Figure 1262 .

54.3.3.11 PSI5-S Clock Divider “3” for WatchDog and DownStream Data Shift Register Trigger Counters . Used for Channel7 (PS_WD_DDTRIG_CLKDIV3)

Figure 1226. PSI5-S Clock Divider “3” for WD and DDSR trigger counters (PS_WD_DDTRIG_CLKDIV3)

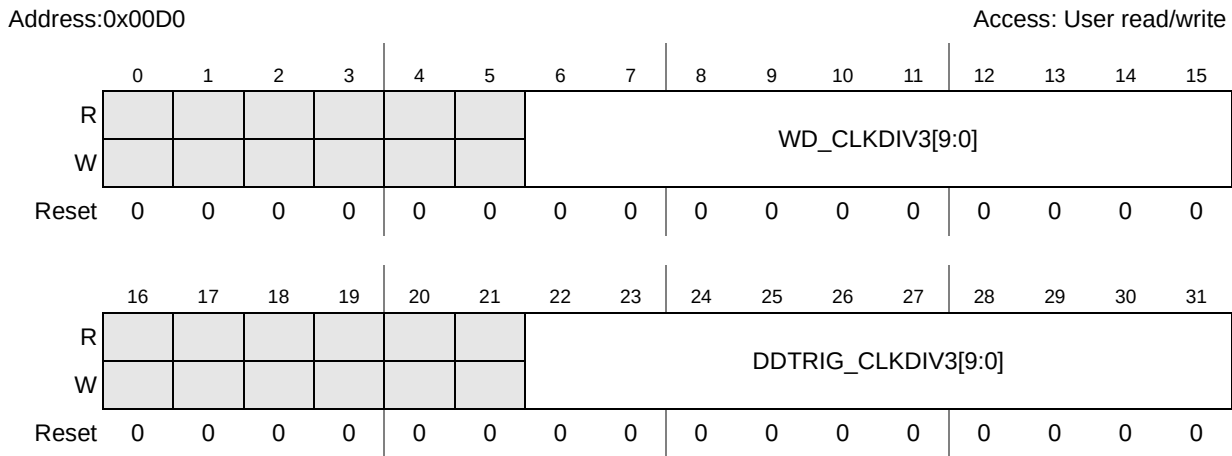


Table 1205. PS_WD_DDTRIG_CLKDIV2 bit field details

Field	Description
WD_CLKDIV3[9:0]	Clock Divider for watchdog clock . This register controls the division factor of the watchdog clock (ipg_clk_ps_wd, please see Figure 1258) in steps of 1us . This register controls the watchdog clock for Channel 7. Also please see Figure 1262 .
DDTRIG_CLKDIV3[9:0]	Clock Divider for DDSR shift trigger generation counters, which in turn are used for sync pulse generation . This register controls the division factor of the DDSR trigger clock(ipg_clk_ps_ddtrig , please see Figure 1253) in steps of 1us . This register controls this clock for Channel 7 . Also please see Figure 1262 .

54.3.3.12 PSI5-S Mbox Status Irq (PS_MBOX_SR_IRQ)

Figure 1227. PSI5-S Mbox Status Irq (PS_MBOX_SR_IRQ)

Address: 0x00E0

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	MBOX_CH7	MBOX_CH6	MBOX_CH5	MBOX_CH4	MBOX_CH3	MBOX_CH2	MBOX_CH1	MBOX_CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1206. PS_MBOX_SR_IRQ bit field details⁽¹⁾

Field	Description
Reserved	Reserved
MBOX_CH7	When set it Indicates at least one message in mailbox 7 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH7[F(0-5)_READ] bits i.e. MBOX_CH7 = F0_READ OR F1_READ ORF5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel7 are cleared.
MBOX_CH6	When set it Indicates at least one message in mailbox 6 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH6[F(0-5)_READ] bits i.e. MBOX_CH6 = F0_READ OR F1_READ ORF5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel6 are cleared.
MBOX_CH5	When set it Indicates at least one message in mailbox 5 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH5[F(0-5)_READ] bits i.e. MBOX_CH5 = F0_READ OR F1_READ ORF5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel5 are cleared.
MBOX_CH4	When set it Indicates at least one message in mailbox 4 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH4[F(0-5)_READ] bits i.e. MBOX_CH4 = F0_READ OR F1_READ ORF5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel4 are cleared.

Table 1206. PS_MBOX_SR_IRQ bit field details⁽¹⁾

Field	Description
MBOX_CH3	When set it Indicates at least one message in mailbox 3 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH3[F(0-5)_READ] bits i.e. MBOX_CH3 = F0_READ OR F1_READ ORF5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel3 are cleared.
MBOX_CH2	When set it Indicates at least one message in mailbox 2 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH2[F(0-5)_READ] bits i.e. MBOX_CH2 = F0_READ OR F1_READ ORF5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel2 are cleared.
MBOX_CH1	When set it Indicates at least one message in mailbox 1is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH1[F(0-5)_READ] bits i.e. MBOX_CH1 = F0_READ OR F1_READ ORF5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel1 are cleared.
0 MBOX_CH0	When set it Indicates at least one message in mailbox 0is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH0[F(0-1)_READ] bits i.e. MBOX_CH0 = F0_READ OR F1_READ . Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel0 are cleared.

1. Also please see [Figure 1259](#).

54.3.3.13 PSI5-S Error Status Irq (PS_ERR_SR_IRQ)

Figure 1228. PSI5-S Error Status Irq (PS_ERR_SR_IRQ)

Address:0x00E4

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	R_UVL_ERR	N_ERR	R_OVL_ERR	F_WD_ERR	SCI_O_ERR	SCI_F_ERR	SCI_P_ERR	HD_ERR	CRC_ERR/ P_ERR	XCRC_ERR
W							W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1207. PS_ERR_SR_IRQ bit field details⁽¹⁾

Field	Description
Reserved	Reserved
R_UVL_ERR	When set indicates that message underflow error(number of UART bytes recieved are less than 3) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[R_UVL_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[R_UVL_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.
N_ERR	When set indicates that N_ERR (Number of UART bytes are not equal to PS_MSGA_CHn[Fn_byte]) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[N_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[N_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.
R_OVL_ERR	When set indicates that message Overflow error(number of UART bytes recieved have exceeded 6) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[R_OVL_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[R_OVL_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.

Table 1207. PS_ERR_SR_IRQ bit field details⁽¹⁾

Field	Description
F_WD_ERR	When set indicates that Frame Watchdog error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[F_WD_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[F_WD_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.
SCI_O_ERR	When set indicates that UART Overrun error(from the UART side) is present in the message currently present in MRU buf2. When "1", then this bit indicates that some UART byte belonging to the concerned UART packet, has been overwritten by another UART byte and is lost. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_O_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_O_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.
SCI_F_ERR	When set indicates that UART Framing error is present in the message currently present in MRU buf2. If this bit is set then it indicates to the application that the synchronisation of the UART module is lost and any subsequent UART bytes recieved will be error prone. In this case the application is required to reset the UART module by using the PS_GCR[SR] bit. Subsequently it should restart the transmission. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_F_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_F_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.

Table 1207. PS_ERR_SR_IRQ bit field details⁽¹⁾

Field	Description
SCI_P_ERR	When set indicates that UART parity error is present in the message currently present in MRU buf2. This bit will be set when there is a parity error in any UART byte using which the concerned UART packet was formed. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_P_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_P_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.
HD_ERR	When set indicates that Header error(E0/E1) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[HD_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[HD_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.

Table 1207. PS_ERR_SR_IRQ bit field details⁽¹⁾

Field	Description
CRC_ERR/P_ERR	When set indicates that PSI5 CRC/Parity error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[CRC_ERR/P_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[CRC_ERR/P_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode. <i>Note: This bit always remains "0" for messages having CID and FID both as "0".</i>
XCRC_ERR	When set indicates that XCRC error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[XCRC_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[XCRC_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. Please see ^(Note:) below. The bit is also writable in PS_NORMAL mode.

1. Also please see [Figure 1259](#).

Note: *.If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.*

54.3.3.14 PSI5-S Mailbox select IRQ[irq_n](PS_MBOX_SEL_IRQ[irq_n]), irq_n=0 to 7

Figure 1229. PSI5-S Mailbox select IRQ[irq_n](PS_MBOX_SEL_IRQ[irq_n])

Address: 0x00E8+hex(8*[irq_n])

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	MBOX_CH7_SEL	MBOX_CH6_SEL	MBOX_CH5_SEL	MBOX_CH4_SEL	MBOX_CH3_SEL	MBOX_CH2_SEL	MBOX_CH1_SEL	MBOX_CH0_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1208. PS_MBOX_SEL_irq[irq_n] bit field details⁽¹⁾

Field	Description
Reserved	Reserved
MBOX_CH7_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_MBOX_SR_IRQ[MBOX_CH7] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH7] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
MBOX_CH6_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_MBOX_SR_IRQ[MBOX_CH6] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH6] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
MBOX_CH5_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_MBOX_SR_IRQ[MBOX_CH5] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH5] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
MBOX_CH4_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_MBOX_SR_IRQ[MBOX_CH4] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH4] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
MBOX_CH3_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_MBOX_SR_IRQ[MBOX_CH3] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH3] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode

Table 1208. PS_MBOX_SEL_irq[irq_n] bit field details⁽¹⁾

Field	Description
MBOX_CH2_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_MBOX_SR_IRQ[MBOX_CH2] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH2] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
MBOX_CH1_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_MBOX_SR_IRQ[MBOX_CH1] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH1] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
MBOX_CH0_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_MBOX_SR_IRQ[MBOX_CH0] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH0] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode

1. Also please see [Figure 1259](#).

54.3.3.15 PSI5-S Error Select IRQ[iq_n] (PS_ERR_SEL_IRQ[irq_n]) irq_n=0 to 7

Figure 1230. PSI5-S Error Enable irq[irq_n] (PS_ERR_SEL_IRQ[irq_n])

Address: 0x00EC+hex(8*[irq_n])

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0										
W							R_UVL_ERR_SEL	N_ERR_SEL	R_OVL_ERR_SEL	F_WD_ERR_SEL	SCI_O_ERR_SEL	SCI_F_ERR_SEL	SCI_P_ERR_SEL	HD_ERR_SEL	CRC_ERR_SEL	XCRC_ERR_SEL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1209. PS_ERR_SEL_IRQ[irq_n] bit field details⁽¹⁾

Field	Description
Reserved	Reserved
R_UVL_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[R_UVL_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[R_UVL_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
N_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[N_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[N_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
R_OVL_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[R_OVL_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[R_OVL_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
F_WD_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[F_WD_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[F_WD_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
SCI_O_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[SCI_O_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_O_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
SCI_F_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[SCI_F_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_F_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
SCI_P_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[SCI_P_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_P_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
HD_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[HD_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[HD_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode

Table 1209. PS_ERR_SEL_IRQ[irq_n] bit field details⁽¹⁾

Field	Description
CRC_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[CRC_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[CRC_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode
XCRC_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: 0 PS_ERR_SR_IRQ[XCRC_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[XCRC_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode

1. Also please see [Figure 1259](#)

54.3.3.16 PSI5-S Watchdog Error Status and Watchdog Timestamp status register (PS_WDGTSSR)

Figure 1231. PSI5-S channel 0 message configuration register A(PS_WDGTSSR)

Address:0x016C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F_WD_ERR_STATUS[7:1]								WDGTS_STATUS[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDGTS_STATUS[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1210. PS_WDGTSSR bit field details

Field	Description
F_WD_ERR_STATUS[7:1]	These bits give the watchdog error status for the watchdog of each corresponding channel . Thus F_WD_ERR_STATUS[1] will give the watchdog error status for watchdog of channel1 . Similarly F_WD_ERR_STATUS[2] will give the watchdog error status for watchdog of channel2 and so on till F_WD_ERR_STATUS[7] , which gives the watchdog error status for watchdog of channel7. As soon as the watchdog of the corresponding channel has a watchdog overrun error, the error status is latched in the corresponding bits of this register. <i>Note: These bits are automatically cleared by the hardware when any of the bits of PS_WDGTSSR is read by the application.</i>
WDTS_STATUS[23:0]	Timestamp value of that specific channel , the timestamp of which is captured when the watchdog error of that specific channel has occurred. This register always contains the timestamp value corresponding to the last occurring Watch Dog error. If due to bad programming watchdog error of more than one channel happen simultaneously then this register contains the timestamp value of the highest channel. For each channel , whether Time Stamp A or B is captured depends on the value of PS_MSGA_CHn[TIMESTAMP_A_B_SEL] register bit. <i>Note: These bits are automatically cleared by the hardware when any of the bits of PS_WDGTSSR is read by the application.</i>

Note: All bits of PS_WDGTSSR register are self clearing on read . This implies that in case this register is read by the application before the corresponding PS_WDGTSSR[F_WD_ERR_STATUS[CHn]] has had a chance to be sent in the next upcoming PSI5 message, then this bit would be sent as a “0” in this upcoming PSI5 message.

54.3.3.17 PSI5-S channel 0 message configuration register A (PS_MSGA_CH0)

Figure 1232. PSI5-S channel 0 message configuration register A(PS_MSGA_CH0)

Address:0x016C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	F0_BYTE[2:0]				0	0	TIME_STAMP_A_B_SEL	0	TSBUF_CLR	TSBUF_EN	0	0
W																	
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 1211. PS_MSGA_CH0 bit field details

Field	Description
Reserved	Reserved
F0_BYTE[2:0]	Number of UART bytes comprising one UART packet. Applicable only to special messages having CID = 0 in their Message Header. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled and a message with the corresponding frame ID is not expected to occur. Please see Fn_byte recalculation that explains the recalculation of Fn_bytes.
Reserved	Reserved
TIMESTAMP_A_B_SEL	Whether the Timestamp corresponding to TimeStamp Counter A or B, is appended to messages targetted for Channel0: 0 Timestamp counter A 1 Timestamp counter B
Reserved	Reserved
TSBUF_CLR	This bit is used to clear the local timestamp buffer for the respective channel. Once cleared the buffer remains clear till the event to capture time stamp is triggered. 1 Timestamp buffer is cleared. This bit is one shot and is always read as "0".
TSBUF_EN	This bit controls whether timestamp buffer maintains a value "0" or is refreshed on a timestamp "capture event" 0 Timestamp buffer maintains value "0" 1 Timestamp buffer responds to a "capture event" and captures the respective timestamp value.
Reserved	Reserved

54.3.3.18 PSI5-S channel 0 message configuration register B (PS_MSGB_CH0)

Figure 1233. PSI5-S channel 0 message configuration register B(PS_MSGB_CH0)

Address: 0x0170

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	F0_payload[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1212. PS_MSGB_CH0 bit field details

Field	Description
Reserved	Reserved
F0_payload[4:0]	Number of bits in payload region of the message (only for channels having CID as "0" in the message header). This channel is always asynchronous. Legal values are between 8 and 28 . Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.

54.3.3.19 PSI5-S Mailbox status register channel0(PS_MBOX_SR_CH0)

Figure 1234. PSI5-S Channel 0 Mailbox ID (PS_MBOX_SR_CH0)

Address:0x0178

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R											F1_ERR	F1_OV	F1_READ	F0_ERR	F0_OV	F0_READ
W	0	0	0	0	0	0	0	0	0	0	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1213. PS_MBOX_SR_CH0 bit field details

Field	Description
Reserved	Reserved
F1_ERR ⁽¹⁾	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0,LOC1) by the DMA , has one or more _ERR bits set . 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0,LOC1) by the DMA , has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register . It is updated when an unrecoverable message(with the CID = 0 and FID=1) is read from the MRU_BUF2 . It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC1. This bit is also writeable in ps_normal mode.

Table 1213. PS_MBOX_SR_CH0 bit field details (continued)

Field	Description
F1_OV	1 Indicates that there is an unread message stored in MBOX0,LOC1 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX0,LOC1. It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC1. This bit is also writeable in ps_normal mode.
F1_READ	MBOX0,LOC1 is reserved for any unrecoverable message(XCRC_ERR has occurred or it is an "illegal message") . It can correspond to any channel from 1 to 7 . 1It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX0,LOC1 by the DMA. 0 No unread message in MBOX0,LOC1. It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC1. This bit is also writeable in ps_normal mode.
F0_ERR ⁽¹⁾	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0,LOC0) by the DMA , has one or more _ERR bits set . 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0,LOC0) by the DMA , has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register . It is updated when a transceiver message(with the CID = 0 and FID=0) is read from the MRU_BUF2 . It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC0. This bit is also writeable in ps_normal mode.
F0_OV	1 Indicates that there is an unread message stored in MBOX0,LOC0 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA . 0 No overwrite of unread message in MBOX0,LOC0. It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC0. This bit is also writeable in ps_normal mode.
F0_READ	MBOX0,LOC0 is reserved for a transceiver message . 1It indicates that the message that has just been read from PS_MRU_BUF2 will be stored in MBOX0,LOC0 by the DMA . 0 No unread message in MBOX0,LOC0. It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC0. This bit is also writeable in ps_normal mode.

1. Also please see [Figure 1259](#).

54.3.3.20 PSI5-S channel message configuration register A (PS_MSGA_CHn) [n=1..7]

Figure 1235. PSI5-S channel message configuration register A (PS_MSGA_CHn)

Address: 0x0190 + hex([n-1]*0x3C) where "n" is Channel Number

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	L_PC_EN	F5_BYTE[2:0]			L_PC5	F4_BYTE[2:0]			L_PC4	F3_BYTE[2:0]			L_PC3	F2_BYTE[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	L_PC2	F1_BYTE[2:0]			L_PC1	F0_BYTE[2:0]			L_PC0	MODE	TIME_STAMP_A_B_SEL	TMSG_TCMD	TSBUF_CLR	TSBUF_EN	G_PC	CH_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1214. PS_MSGA_CHn bit field details

Field	Description
L_PC_EN	Selects whether the parity/CRC selection for a PSI5 message is to be controlled globally on a channel basis (through the G_PC bits) or locally on a per frame basis (through the L_PC[frame] bits) 0 Parity/CRC option controlled globally through the G_PC bit 1 Parity/CRC option controlled locally through the L_PC[frame] bits individually for message of each frame.
F5_BYTE[2:0]	Number of bytes in UART packet for message corresponding to frame 5. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F5_payload] is immaterial. Please see Fn_byte recalculation that explains the recalculation of Fn_bytes.
L_PC5	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame5 1 CRC selected for PSI5 message for frame5.
F4_BYTE[2:0]	Number of bytes in UART packet for message corresponding to frame 4. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F4_payload] is immaterial. Please see Fn_byte recalculation that explains the recalculation of Fn_bytes.
L_PC4	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame4 1 CRC selected for PSI5 message for frame4

Table 1214. PS_MSGA_CHn bit field details (continued)

Field	Description
F3_BYTE[2:0]	Number of bytes in UART packet for message corresponding to frame 3. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F3_payload] is immaterial. Please see Fn_byte recalculation that explains the recalculation of Fn_bytes.
L_PC3	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame3 1 CRC selected for PSI5 message for frame3.
F2_BYTE[2:0]	Number of bytes in UART packet for message corresponding to frame 2. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F2_payload] is immaterial. Please see Fn_byte recalculation that explains the recalculation of Fn_bytes.
L_PC2	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame2 1 CRC selected for PSI5 message for frame2.
F1_BYTE[2:0]	Number of bytes in UART packet for message corresponding to frame 1. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled (unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F1_payload] is immaterial. Please see Fn_byte recalculation that explains the recalculation of Fn_bytes.
L_PC1	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame1 1 CRC selected for PSI5 message for frame1
F0_BYTE[2:0]	Number of bytes in UART packet for message corresponding to frame 0. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F0_payload] is immaterial. Please see Fn_byte recalculation that explains the recalculation of Fn_bytes.
L_PC0	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame0 1 CRC selected for PSI5 message for frame0.
MODE	This bit selects the operating modes between sync and async for the channel. 1 Asynchronous 0 Synchronous
TIME_STAMP_A_B_SEL	Decides whether Timestamp Counter A or B, value is captured in the local timestamp buffer, whenever a "capture event" occurs: 0 Timestamp counter A 1 Timestamp counter B.

Table 1214. PS_MSGA_CHn bit field details (continued)

Field	Description
TMSG_TCMD	Whether the timestamp is recorded when the message is received or when the channel command is moved from the TxFIFO to the UART Tx register (emulates a sync pulse) 1 Timestamp recorded based on message header byte received 0 Timestamp recorded based on “emulated” sync pulse.
TSBUF_CLR	This bit is used to clear the local timestamp buffer for the respective channel. Once cleared the buffer remains clear till the event to capture time stamp is triggered. 1 TimeStamp buffer is cleared. This bit is one shot and is always read as “0”.
TSBUF_EN	This bit controls whether timestamp buffer maintains a value “0” or is refreshed on a timestamp “capture event” 0 Timestamp buffer maintains value “0” 1 Timestamp buffer responds to a “capture event” and captures the respective timestamp value.
G_PC	This bit globally selects the Parity or the CRC option for all the Frames of that particular channel provided L_PC_EN is set “0” 0 Parity option selected globally for messages of all frames. 1 CRC option selected globally for messages of all frames.
CH_EN	PSI5 Channel Enable. This bit decides if the IP takes the data belonging to the specific channel as “legal” or “illegal”. 0: Concerned PSI5 channel is disabled. Any data occurring with a CID value of a disabled channel is treated as an “illegal message” and send to Channel 0 unrecoverable message area (MBOX ₀ , LOC ₁) 1: Concerned PSI5 channel is enabled. Any data occurring with a CID value of an enabled channel can either be treated as “legal” or “illegal”. Please see Section 54.4.2.1, PSI5-S Message extraction for the definition of legal and illegal messages. Legal messages are sent to MBOX _{CID} , LOC _{FID} while illegal messages are sent to MBOX ₀ , LOC ₁

54.3.3.21 PSI5-S channel message configuration register B (PS_MSGB_CHn)[n=1..7]

Figure 1236. PSI5-S channel message configuration register B (PS_MSGB_CHn)

Address: 0x0194 + hex([n-1]*0x3C) where “n” is Channel Number

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	F5_payload[4:0]				F4_payload[4:0]				F3_payload[4:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F2_payload[4:0]				F1_payload[4:0]				F0_payload[4:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1215. PS_MSGB_CHn bit field details

Field	Description
Reserved	Reserved
F5_payload[4:0]	Number of bits in the Payload region of the PSI5 message corresponding to frame 5. Legal values are between 8 and 28 . Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
F4_payload[4:0]	Number of bits in the Payload region of the PSI5 message corresponding to frame 4. Legal values are between 8 and 28 . Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
F3_payload[4:0]	Number of bits in the Payload region of the PSI5 message corresponding to frame 3. Legal values are between 8 and 28 . Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
F2_payload[4:0]	Number of bits in the Payload region of the PSI5 message corresponding to frame 2. Legal values are between 8 and 28 . Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
F1_payload[4:0]	Number of bits in the Payload region of the PSI5 message corresponding to frame 1. Legal values are between 8 and 28 . Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
F0_payload[4:0]	Number of bits in the Payload region of the PSI5 message corresponding to frame 0. Legal values are between 8 and 28 . Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.

Note: *Fn_byte recalculation. The IP expects legal values to be programmed for Fn_byte field. If this is not followed then the IP reprogrammes the registers with default values. Whenever the value of “Fn_byte” field becomes 1 or 2 it is defaulted to 3 and value of 7 is defaulted to 6. This updation happens once the IP enters the PS_NORMAL mode.*

In addition to the above , the following checks are also performed.

By default the IP will expect that there is coherence between the expected number of bytes calculated through the “minUARTbytes” and the number of bytes in the “Fn_byte” field . Please see [Figure 1246](#) for definition of “minUARTbytes”. If there is no coherence (i.e. the above two calculations do not match) then the IP does the following recalculation :

When Programmed Fn_byte > “minUARTbytes” , then the IP will expect that the additional extra bytes would be stuffed with 0s from the transceiver. Rest of the checking will be done

as described in [Section 54.4.2.2, PSI5-S “Recoverable and Unrecoverable” Messages](#).

When $F_n_byte < \text{“minUARTbytes”}$ then the “ F_n_bytes ” is internally “increased” such that the expanded “ F_n_bytes ” is now equal to minUARTbytes . Please see [Figure 1246](#) for definition of “ minUARTbytes ”. Using this expanded “ F_n_bytes ” value the message that would be arriving is analyzed. This would mean that no data bits are lost. Rest of the checking will be done as described in [Section 54.4.2.2, PSI5-S “Recoverable and Unrecoverable” Messages](#).

The “recalculated” F_n_byte field is automatically updated by the IP once the IP enters the NORMAL mode of operation. For the subsequent occurrence in the document, all references to “ F_n_byte ” would mean the recalculated value of “ F_n_byte ”.

54.3.3.22 PSI5-S Mailbox status register channel (PS_MBOX_SR_CHn) [n=1..7]

Figure 1237. PSI5-S Channel Mailbox ID (PS_MBOX_SR_CHn)

Address: $0x019C + \text{hex}([n-1]*0x3C)$ where “n” is Channel Number

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F5_ERR	F5_OV
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F5_READ	F4_ERR	F4_OV	F4_READ	F3_ERR	F3_OV	F3_READ	F2_ERR	F2_OV	F2_READ	F1_ERR	F1_OV	F1_READ	F0_ERR	F0_OV	F0_READ
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1216. PS_MBOX_SR_CHn bit field details

Field	Description
Reserved	Reserved
F5_ERR ⁽¹⁾	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC5) by the DMA, has one or more _ERR bits set . 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC5) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register . It is updated when the corresponding message(with the CID = n and FID=5) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC5. This bit is also writeable in ps_normal mode.
F5_OV	1 Indicates that there is an unread message stored in MBOX[n],LOC5 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX[n],LOC5. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC5. This bit is also writeable in ps_normal mode.
F5_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC5. 0 No unread message in MBOX[n],LOC5. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC5. This bit is also writeable in ps_normal mode.
F4_ERR ⁽¹⁾	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC4) by the DMA , has one or more _ERR bits set . 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC4) by the DMA , has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register . It is updated when the corresponding message(with the CID = n and FID=4) is read from the MRU_BUF2 by the DMA . It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC4. This bit is also writeable in ps_normal mode.
F4_OV	1 Indicates that there is an unread message stored in MBOX[n],LOC4 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX[n],LOC4. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC4. This bit is also writeable in ps_normal mode.

Table 1216. PS_MBOX_SR_CHn bit field details (continued)

Field	Description
F4_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC4. 0 No unread message in MBOX[n],LOC4. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC4. This bit is also writeable in ps_normal mode.
F3_ERR ⁽¹⁾	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC3) by the DMA, has one or more _ERR bits set . 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC3) by the DMA, has ALL the “_ERR” bits as “0”. This bit is the “OR” of the contents of the PS_MBOX_SR_IRQ register . It is updated when the corresponding message(with the CID = n and FID=3) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC3. This bit is also writeable in ps_normal mode.
F3_OV	1 Indicates that there is an unread message stored in MBOX[n],LOC3 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA . 0 No overwrite of unread message in MBOX[n],LOC3. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC3. This bit is also writeable in ps_normal mode.
F3_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC3 by the DMA . 0 No unread message in MBOX[n],LOC3. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC3. This bit is also writeable in ps_normal mode.
F2_ERR ⁽¹⁾	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC2) by the DMA , has one or more _ERR bits set . 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC2) by the DMA, has ALL the “_ERR” bits as “0”. This bit is the “OR” of the contents of the PS_MBOX_SR_IRQ register . It is updated when the corresponding message(with the CID = n and FID=2) is read from the MRU_BUF2 by the DMA . It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC2. This bit is also writeable in ps_normal mode.

Table 1216. PS_MBOX_SR_CHn bit field details (continued)

Field	Description
F2_OV	1 Indicates that there is an unread message stored in MBOX[n],LOC2 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA . 0 No overwrite of unread message in MBOX[n],LOC2. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC2. This bit is also writeable in ps_normal mode.
F2_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC2. 0 No unread message in MBOX[n],LOC2. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC2. This bit is also writeable in ps_normal mode.
F1_ERR ⁽¹⁾	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC1) by the DMA, has one or more _ERR bits set . 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC1) by the DMA, has ALL the “_ERR” bits as “0”. This bit is the “OR” of the contents of the PS_MBOX_SR_IRQ register . It is updated when the corresponding message(with the CID = n and FID=1) is read from the MRU_BUF2 by the DMA . It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC1. This bit is also writeable in ps_normal mode.
F1_OV	1 Indicates that there is an unread message stored in MBOX[n],LOC1 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA . 0 No overwrite of unread message in MBOX[n],LOC1. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC1. This bit is also writeable in ps_normal mode.
F1_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC1. 0 No unread message in MBOX[n],LOC1. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC1. This bit is also writeable in ps_normal mode.

Table 1216. PS_MBOX_SR_CHn bit field details (continued)

Field	Description
F0_ERR ⁽¹⁾	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC0) by the DMA , has one or more _ERR bits set . 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC0) by the DMA , has ALL the “_ERR” bits as “0”. This bit is the “OR” of the contents of the PS_MBOX_SR_IRQ register . It is updated when the corresponding message(with the CID = n and FID=0) is read from the MRU_BUF2 by the DMA . It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC0. This bit is also writeable in ps_normal mode.
F0_OV	1 Indicates that there is an unread message stored in MBOX[n],LOC0 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA . 0 No overwrite of unread message in MBOX[n],LOC0. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC0. This bit is also writeable in ps_normal mode.
F0_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC0. 0 No unread message in MBOX[n],LOC0. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC0. This bit is also writeable in ps_normal mode.

1. Also please see [Figure 1259](#).

54.3.3.23 PSI5-S channel watchdog configuration register (PS_WD_CFGR_CHn) [n==1..7]

Figure 1238. PSI5-S channel watchdog configuration register (PS_WD_CFGR_CHn)

Address: 0x01A0 + hex([n-1]*0x3C) where "n" is Channel Number

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	WDRST	WDCS	WDEN	0	0	0	0	0	0	0	0	0
W					WDRST	WDCS	WDEN		WD_TO[23:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WD_TO[15:0]															
W	WD_TO[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1217. PS_WD_CFGR_CHn bit field details

Field	Description
WDRST	Resets watchdog counter for the channel This bit is one shot and is also writable in ps_normal mode. It is always read as 0. It will reset the watchdog counter and the counter will be reloaded with the value of WD_TO and automatically start counting again.
WDCS	This will select either GTM or module clock for watchdog counter of the channel. Please see Figure 1262 . 0 ipg_clk_ps_wd Clock 1 gtm_trig is selected as the clock.
WDEN	The watchdog counter for each channel can be started, stopped. 1 watchdog counter started 0 watchdog counter stopped. It maintains its value. This bit is also writable in ps_normal mode

Table 1217. PS_WD_CFGR_CHn bit field details (continued)

Field	Description
Reserved	Reserved
WD_TO[23:0]	This indicates the watchdog time out value . The watchdog generates an overrun error, in case the watchdog is not refreshed before the watchdog counter reached the WD_TO value. The Watchdog is refreshed in a different manner depending on whether the PSI5-S channel is in asynchronous mode or synchronous mode as described below : In Asynchronous mode : Whether the WD has to be “refreshed” or “not to be refreshed” is analyzed when the XCRC is calculated . In case the XCRC is correct the watchdog is refreshed(It is not STOPPED) , else the WD counter keeps on running normally eventually leading to a watchdog overrun error. In Synchronous Mode: The watchdog is ALWAYS refreshed on the occurrence of the internal sync pulse and keeps on counting thereafter. However, it is stopped when the “Packet_counter” reaches the count of “total number of configured slots”. The “Packet_counter” here is a counter that keeps track of the number of configured(enabled) slots. It is incremented on the reception of each UART packet with a “CID” and “FID” such that PS_MSGA_CH_CID[FID_byte] is not equal to “0” . The packet counter is however incremented ONLY if the XCRC of the received UART packet is correct else it is not incremented. In case the “Packet Counter” does not reach the “expected packet count” within the “WD timeout configured “ , the WD overrun error is generated. In both the above modes , the watchdog is always refreshed on the occurrence of Watchdog overrun error (F_WD_ERR) or a software reload . <i>Note: The watchdog counter is enabled ONLY if the channel is enabled.</i>

54.3.3.24 PSI5-S DDSR Trigger offset register channel(PS_DDTRIG_OFFR_CHn)[n=1..7]

Figure 1239. PSI5-S DDSR Trigger offset register channel (PS_DDTRIG_OFFR_CHn)

Address: 0x01A4 + hex([n-1]*0x3C) where “n” is Channel Number

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DDTRIG_OFFR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1218. PS_DDTRIG_OFFR_CHn

Field	Description
Reserved	Reserved
DDTRIG_OFFR[15:0]	These bits configure the delay between the “shift trigger” of the various DDSRs across different channels. The value in this register governs the delay after which the specific DDSR would be triggered once the Global Shift trigger (PS_GLCR[GL_DDSR_TRIG]) is written to “1” and the corresponding PS_E2SCR_CH[n][GL_TRIG_SEL] is set to “1” for that channel. There is an internal counter which is clocked on the “ipg_clk_ps_ddtrig” clock . The counters count up to the value as programmed in the PS_DDTRIG_PERR_CH[n] registers after which they are reset. At each reset a shift trigger is generated for the DDSR. The “start” of these counters is delayed based on the value programmed in the PS_DDTRIG_OFFR_CH[n] register

54.3.3.25 PSI5-S DDSR Trigger period register channel(PS_DDTRIG_PERR_CHn)[n=1..7]

Figure 1240. PSI5-S DDSR Trigger period register channel (PS_DDTRIG_PERR_CHn)

Address: 0x01A8 + hex([n-1]*0x3C) where “n” is Channel Number

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DDTRIG_PERR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1219. PS_DDTRIG_PERR_CHn

Field	Description
Reserved	Reserved
DDTRIG_PERR[15:0]	These bits configure the “shift trigger” period for the DDSR of that specific channel. The value in this register governs the period with which the data for the DDSR of that particular channel would be shifted out. There is an internal counter which is clocked on the “ipg_clk_ps_ddtrig” clock . The counters count up to the value as programmed in the PS_DDTRIG_PERR_CH[n] registers after which they are reset. At each reset a shift trigger is generated for the DDSR. The “start” of these counters is delayed based on the value programmed in the PS_DDTRIG_OFFR_CH[n] register

54.3.3.26 PSI5-S ECU to Sensor Control Register (PS_E2SCR_CHn)[n=1..7]

Figure 1241. PSI5-S ECU to Sensor Control Register (PS_E2SCR_CHn)

Address: 0x01AC + hex([n-1]*0x3C) where "n" is Channel Number

Access: User read/write

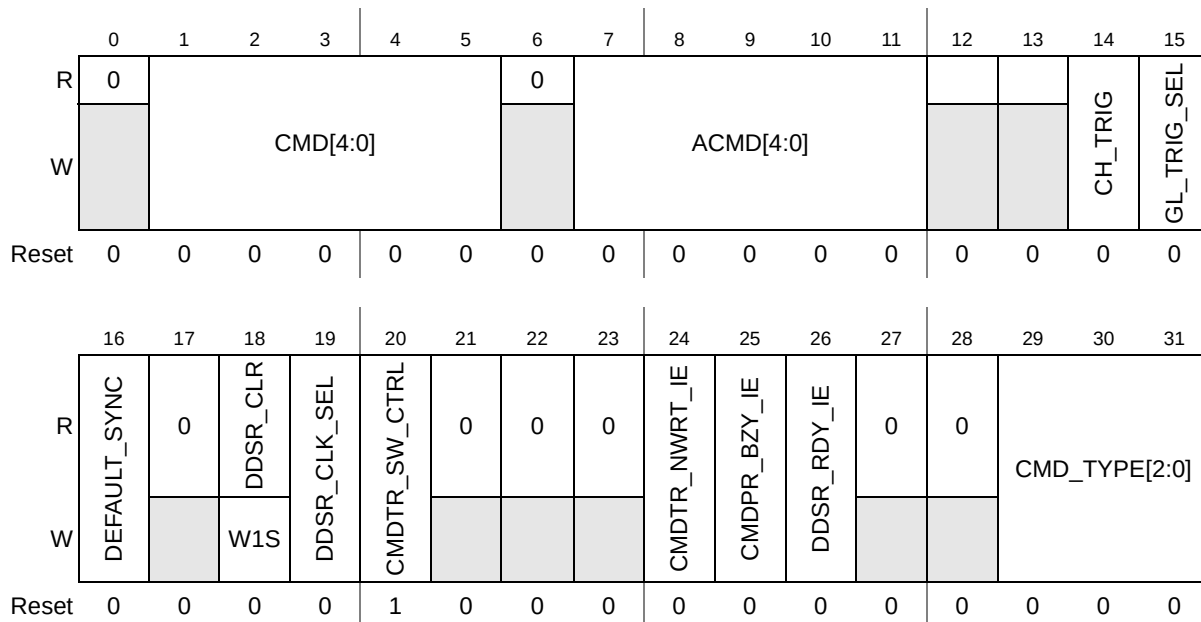


Table 1220. PS_E2SCR_CHn bit details

Field	Description
Reserved	Reserved Always read as 0
CMD[4:0]	Bits for the ECU-to-sensor "CMD" format. They refer to the command send to the UART Tx logic, when a "1" is shifted from the DDSR register <i>Note: This bit is writable only in PS_CONFIG mode.</i>
Reserved	Reserved Always read as 0
ACMD[4:0]	Bits for the ECU-to-sensor "ACMD" format. They refer to the command send to the UART Tx logic, when a "0" is shifted from the DDSR register. <i>Note: This bit is writable only in PS_CONFIG mode.</i>
Reserved	Reserved Always read as 0
CH_TRIG	This controls that the DDSR of the specific channel is triggered locally. This bit controls the shifting when PS_E2SCR_CHn[GL_TRIG_SEL] is 0. 1 Start the "shift trigger" of the DDSR. 0 Stop the DDSR from shifting the commands. The counters corresponding to DDTRIG_PERR and the DDTRIG_OFFR are reset. <i>Note: This bit is also writable in PS_NORMAL mode.</i>

Table 1220. PS_E2SCR_CHn bit details (continued)

Field	Description
GL_TRIG_SEL	This bit controls whether the DDSR shift trigger is a global trigger or a local channel trigger. 1 The DDSR shift trigger is a global trigger dependent on PS_GLCR[GL_DDSR_TRIG] . 0 The DDSR shift trigger is a local channel trigger dependent on the bit CH_TRIG. <i>Note: This bit is writable only in PS_CONFIG mode.</i>
DEFAULT_SYNC	Decides the value to which the DDSR should default to when PS_E2SSR_CHn[DDSR_RDY] == 1. When DDSR_RDY == 1 then the default shift (whether it is CMD or ACMD) is based on the value of this bit. <i>Note: This bit is writable only in PS_CONFIG mode</i>
Reserved	Reserved Always read as 0
DDSR_CLR	Used for rejecting the contents of the DDSR register. As soon as this bit is written to “1” the contents of DDSR are immediately rejected and defaulted to “0” after the completion of the current ACMD or the CMD transmission. PS_E2SSR_CHn[DDSR_RDY] goes to 1 and value of PS_E2SCR_CHn[DEFAULT_SYNC] starts getting shifted. This bit is one shot (W1S). <i>Note: This bit is also writable in PS_NORMAL mode.</i>
DDSR_CLK_SEL	Selects which clock clocks shift logic of the DDSR, for shifting the bits for command transmission: 0 ipg_clk_ps_ddtrig Clock 1 gtm_trig is selected as the clock.
CMDTR_SW_CTRL	This bit decides that once the command is ready for transmission (indicated by the setting PS_E2SSR_CHn[CMDPR_BZY] to 1), then when should the actual transmission start: 0 The actual transmission of command remains suspended, even though the command is ready in the DDSR. 1 The command transmission starts once the PS_E2SSR_CHn[CMDPR_BZY] becomes 1, and the internal shift logic is available. <i>Note: This bit is also writable in PS_NORMAL mode.</i>
Reserved	Reserved Always read as 0
CMDTR_NWRT_IE	Interrupt Enable for DDSR being attempted to be written when it is still not empty i.e PS_E2SSR_CHn[DDSR_RDY] == 0 yet an attempt to write to DDSR is made. 1: Enable the interrupt 0: Disable the interrupt <i>Note: This bit is also writable in PS_NORMAL mode.</i>
CMDPR_BZY_IE	Interrupt Enable for DDSR getting updated with the internally processed complete command (with CRC, stuff bits, start bits getting appended) 1: Enable the interrupt 0: Disable the interrupt <i>Note: This bit is also writable in PS_NORMAL mode.</i>

Table 1220. PS_E2SCR_CHn bit details (continued)

Field	Description																											
DDSR_RDY_IE	Interrupt Enable for DDSR getting empty and ready for receiving new command. 1: Enable the interrupt 0: Disable the interrupt <i>Note: This bit is also writable in PS_NORMAL mode</i>																											
Reserved	Reserved Always read as 0																											
CMD_TYPE[2:0]	<table><tr><td>CMD_TYPE</td><td>PSI5 Frame</td><td>Number of bits writable by the Application in the DDSR</td></tr><tr><td>"000" - 0</td><td>Short Frame with 31 "1s" as the start condition*</td><td>6 LSBs of DDSR_L</td></tr><tr><td>"001" - 1</td><td>Short Frame with 5 "0s" as the start condition*</td><td>6 LSBs of DDSR_L</td></tr><tr><td>"010" - 2</td><td>Long Frame with 31 "1s" as the start condition*</td><td>16 LSBs of DDSR_L</td></tr><tr><td>"011" - 3</td><td>Long Frame with 5 "0s" as the start condition*</td><td>16 LSBs of DDSR_L</td></tr><tr><td>"100" - 4</td><td>X-Long Frame with 31 "1s" as the start condition*</td><td>22 LSBs of DDSR_L</td></tr><tr><td>"101" - 5</td><td>X-Long Frame with 5 "0s" as the start condition*.</td><td>22 LSBs of DDSR_L</td></tr><tr><td>"110" - 6</td><td>XX-Long</td><td>24 LSBs of DDSR_L</td></tr><tr><td>"111" - 7</td><td>Illegal</td><td></td></tr></table> * The start condition(31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command; neither is this visible in any register.	CMD_TYPE	PSI5 Frame	Number of bits writable by the Application in the DDSR	"000" - 0	Short Frame with 31 "1s" as the start condition*	6 LSBs of DDSR_L	"001" - 1	Short Frame with 5 "0s" as the start condition*	6 LSBs of DDSR_L	"010" - 2	Long Frame with 31 "1s" as the start condition*	16 LSBs of DDSR_L	"011" - 3	Long Frame with 5 "0s" as the start condition*	16 LSBs of DDSR_L	"100" - 4	X-Long Frame with 31 "1s" as the start condition*	22 LSBs of DDSR_L	"101" - 5	X-Long Frame with 5 "0s" as the start condition*.	22 LSBs of DDSR_L	"110" - 6	XX-Long	24 LSBs of DDSR_L	"111" - 7	Illegal	
CMD_TYPE	PSI5 Frame	Number of bits writable by the Application in the DDSR																										
"000" - 0	Short Frame with 31 "1s" as the start condition*	6 LSBs of DDSR_L																										
"001" - 1	Short Frame with 5 "0s" as the start condition*	6 LSBs of DDSR_L																										
"010" - 2	Long Frame with 31 "1s" as the start condition*	16 LSBs of DDSR_L																										
"011" - 3	Long Frame with 5 "0s" as the start condition*	16 LSBs of DDSR_L																										
"100" - 4	X-Long Frame with 31 "1s" as the start condition*	22 LSBs of DDSR_L																										
"101" - 5	X-Long Frame with 5 "0s" as the start condition*.	22 LSBs of DDSR_L																										
"110" - 6	XX-Long	24 LSBs of DDSR_L																										
"111" - 7	Illegal																											

54.3.3.27 PSI5-S ECU to Sensor Status Register (PS_E2SSR_CHn)[n=1..7]

Figure 1242. PSI5-S ECU to Sensor Status Register (PS_E2SSR_CHn)

Address: 0x01B0 + hex([n-1]*0x3C) where "n" is Channel Number

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CMDTR_NWRT	CMDPR_BZY	DDSR_RDY	0	0	0	0	0
W									W1C	W1C	W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1221. PS_E2SSR_CHn bit field details

Field	Description
Reserved	Reserved Always read as 0
CMDTR_NWRT	Status of Command written to the DDSR: 1 Command written to the DDSR is rejected as an attempt to write to it has been made when DDSR_RDY == 0. 0 Command written to the DDSR has been accepted for processing.
CMDPR_BZY	Command Processing Busy Status . This bit along with the DDSR_RDY bit can be used to check the status of the command that is being processed in the DDSR. 1 DDSR is ready with the processed command complete with CRC/start bits/stuff bits. The transmission of the same would start if PS_E2SCR_CHn[CMDTR_SW_CTRL] == 1 . For this status DDSR_RDY should also be read as "0" . 0 DDSR is busy with command processing. For this status DDSR_RDY should also be read as "0" . Note that when DDSR_RDY goes to "1" then this bit goes to "0" . So it represents the above status conditions ONLY when DDSR_RDY is "0".
DDSR_RDY	Status for DDSR getting empty and ready for receiving new command. 1 DDSR is empty and ready for receiving a new command. 0:DDSR is not ready and is busy with either the command processing or the command transmission
Reserved	Reserved Always read as 0

54.3.3.28 PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PS_DDSR_H_CHn)[n=1..7]

Figure 1243. PSI5-S channel1 ECU to Sensor Upstream Data Shift Register High (PS_DDSR_H_CHn)

Address: 0x01B4 + hex([n-1]*0x3C) where "n" is Channel Number Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DDSR_H										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1222. PS_DDSR_H_CHn bit details

Field	Description
Reserved	Reserved Always read as 0
DDSR_H	Upper High bits of the 43 bit complete DDSR register. These bits are updated by the hardware and software cannot write to these bits. They can be read by the software to check for the processed command after the command is written to the DDSR by the application , the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to "1".

54.3.3.29 PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PS_DDSR_L_CHn)[n=1..7]

Figure 1244. PSI5-S channel1 ECU to Sensor Upstream Data Shift Register Low (PS_DDSR_L_CHn)

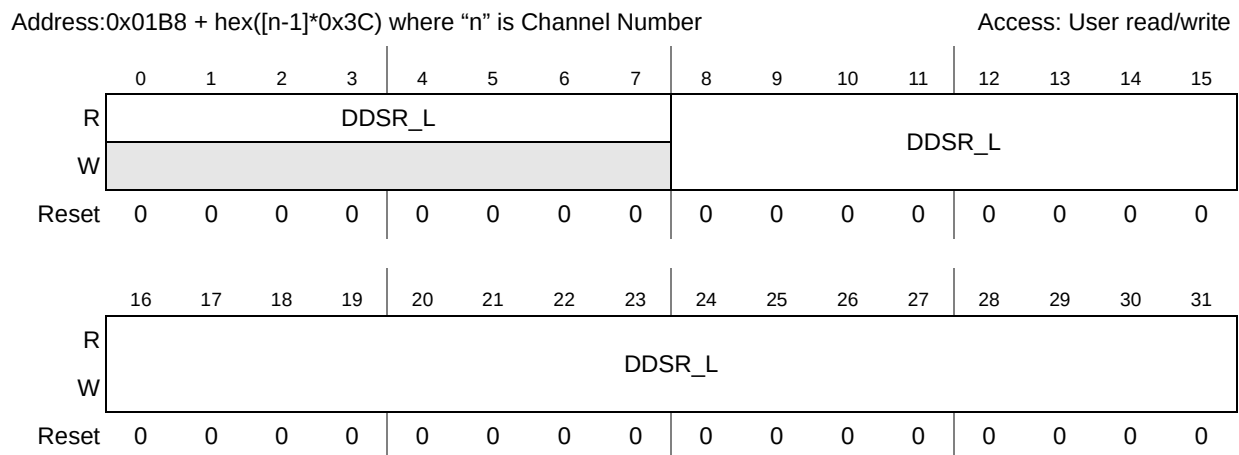


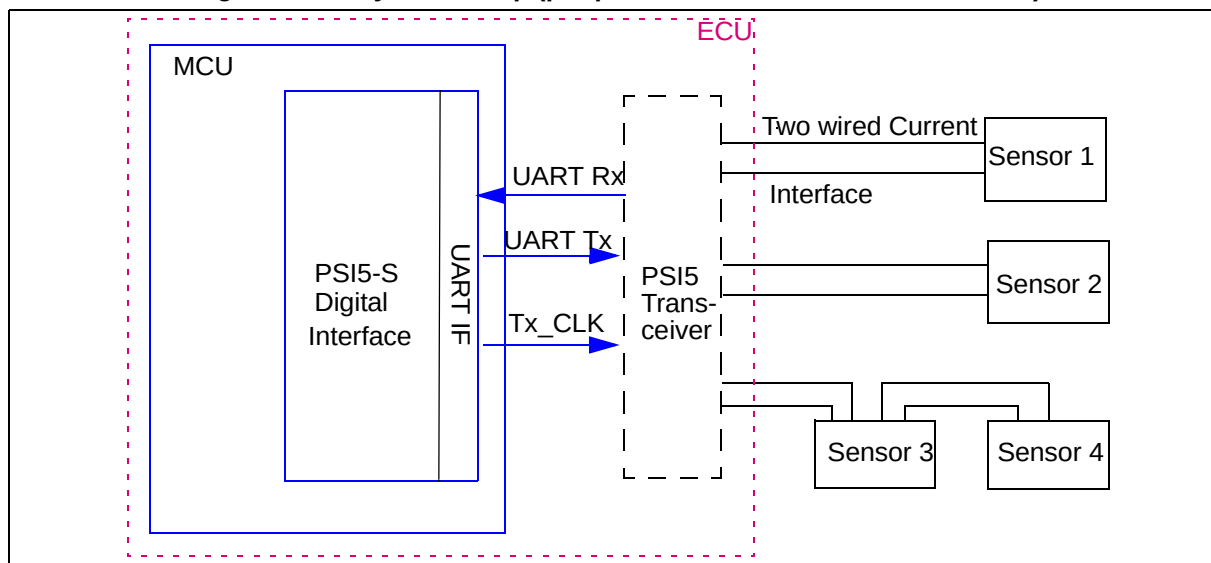
Table 1223. PS_DDSR_L_CHn bit field details

Field	Description
DDSR_L	8 bits of the DDSR_L that can only be read by the Application. These 8 bits are automatically updated by the hardware . They can be read by the software to check for the processed command after the command is written to the DDSR by the application , the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to "1".
DDSR_L	Lower 24 bits of the DDSR register that are written by the Application as well as updated by the hardware once the command processing is over. Though all the 24 bits can be written by the software , yet only those number of bits as governed by PS_E2SSR_CHn[CMD_TYPE] are actually taken for processing. This register can be read by the software to check for the processed command, after the command is written to the DDSR by the application , the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to "1". <i>Note: These bits are also writable in the PS_NORMAL mode.</i>

54.4 Functional description

The PSI5 is a communication protocol for peripheral inertial, pressure, temperature, and position sensors. These sensors are connected to the ECU by just two wires, using the same lines for power supply and data transmission. The UART compatible transceiver (present externally) provides a pre-regulated voltage to the sensors and reads in the transmitted sensor data. [Figure 1245](#) shows a point-to-point connection for sensors 1 and 2 and bus configuration for sensors 3 and 4.

Figure 1245. System setup (peripheral sensor connected to device)



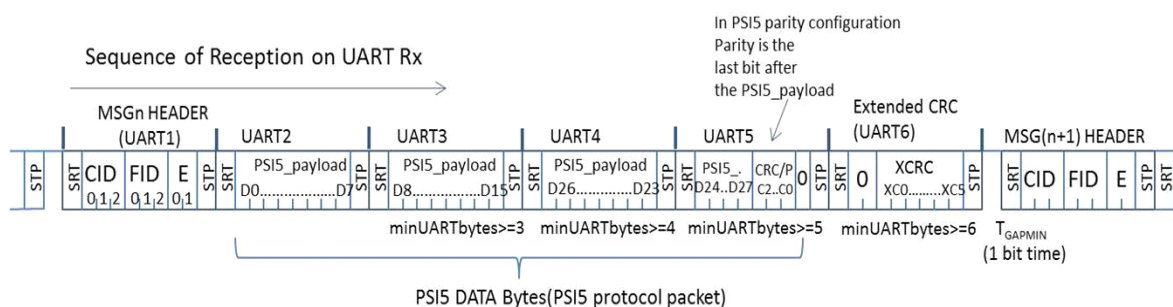
ECU (Electronic Control Unit) is comprised of PSI5-S digital interface and the UART compatible PSI5 transceiver (present externally). The PSI5-S digital interface communicates with the PSI5 Transceiver through the UART interface as shown in [Figure 1245](#). When conducting the Sensor To ECU communication, the external transceiver converts the sensor data to the UART packet format before sending it to the PSI5-S module. The IP extracts the PSI5 Frames from the input UART Packet.

54.4.1 PSI5-S UART Message Structure

The transceiver sends a UART message packet consisting of between three to six UART bytes which contains information to allow rebuilding of the PSI5 message as well as the message itself and has a CRC which protects the complete packet (XCRC).

Each byte is made up of one start bit, eight data bits and one stop bit - no parity is included in the UART byte. [Figure 1246](#) shows the PSI5-S UART packet structure.

Figure 1246. PSI5-S Packet Structure



Total Message Payload(TMPL) = header_bits(8)+PSI5_payload(8-28bits) +CRC/P(3/1 bit)+XCRC(6bits)

minUARTbytes=CEIL(TMPL/8) where CEIL gives the largest integer greater than or equal to TMPL/8

PSI5_payload is the value obtained from PS_MSGA_CHn[Fn_payload] register bits.

XCRC(Extended CRC) can occupy any byte from UART3 to UART6 depending on the number of bytes received.

The UART packet is made up of the following:

- **MSGnHEADER:** This is the message header byte which includes the indication of which channel the PSI5 message was received on (3-bit CID), the slot on that channel in which the message was received (3-bit FID) and transceiver error indication bits (2-bit E)
- **PSI5 DATA bytes** - minimum one, maximum four, depending on the data payload region of the PSI5 message (8 - 28 bits) and whether parity (1-bit) or CRC (3-bit) is configured for the PSI5 message. In the above example shown in [Figure 1246](#) it is taken as 28 bits and 4 UART bytes are shown (UART2-UART5). Messages having CID and FID both as "0s" do not have the CRC/Parity field in them. Such messages only have the message header, payload and the xcrc fields.
- **XCRC- 6-bits** - This is the extended CRC which protects the UART packet. Note that any bits between the end of the PSI5 message payload and the XCRC are 'stuffed' with '0's. This can also be seen in [Figure 1246](#). For details about XCRC handling please refer [Section 54.4.2.5, PSI5-S XCRC handling](#).
- Example to calculate the minUARTbytes (for messages NOT having CID and FID both as 0):
 - $F_n_payload = 14$ with PSI5 CRC option : $TMPL = 8+14+3+6=31$,
 $minUARTbytes=CEIL(3.8)=4$.
 These input data stream is interpreted as
UART1(header),**UART2**(D[0:7]),**UART3**(D[8:13],CRC[2:1]),
UART4(CRC[0],0,XCRC[0:5]).
 - $F_n_payload = 13$ with PSI5 Parity option :
 $TMPL=8+13+1+6=28$, $minUARTbytes=CEIL(3.5)=4$.
 These input data stream is interpreted as
UART1(header),**UART2**(D[0:7]),**UART3**(D[8:12],Paritybit,00),
UART4(00,XCRC[0:5]). Note the "0" stuffing between the parity bit and the XCRC.
 The XCRC is always in the last 6 bits of the last byte.
- Example to calculate the minUARTbytes (for messages having CID and FID both as 0):
 - $F_0_payload = 8$: $TMPL = 8+8+0+6=22$, $minUARTbytes=CEIL(2.75)=3$
 These input data stream is interpreted as
UART1(header),**UART2**(D[0:7]), **UART3**(00,XCRC[0:5]).
 - $F_0_payload = 10$: $TMPL = 8+10+0+6=24$, $minUARTbytes=CEIL(3)=3$
 These input data stream is interpreted as
UART1(header),**UART2**(D[0:7]), **UART3**(D[8:9],XCRC[0:5]).

54.4.2 PSI5-S Received Message handling

The message handling in the IP can be divided in three parts:

- PSI5 message extraction:
This decides when a UART packet has been received with the correct header byte, and then how to extract the information from this UART packet form a complete PSI5 message .
- PSI5 “recoverable” and “unrecoverable messages”:
This differentiates the PSI5 messages received above, into “recoverable” and “unrecoverable” messages.
- PSI5 message storage and Mailbox transfer:
This ensures the transfer of the PSI5 message to the correct location in the system RAM, once the PSI5 message has been correctly differentiated into “recoverable” and “unrecoverable” messages.

Following sections detail the above 3 parts:

54.4.2.1 PSI5-S Message extraction

Once a UART start bit is detected after the idle timeout(the idle time between message packets on the UART interface i.e. the number of bit times between message packets in the range of at least 1 to 16 , configurable in the UART module through the PS_UARTPTO register) has occurred , the UART bytes are subsequently sampled . They are being received by an internal state machine in the MRU, which checks for the formation of a complete UART packet, its coherence and extraction of the PSI5 message from the UART packet . Also please see [Figure 1247](#) for further understanding of this section.Each UART packet that is received can be considered as “recoverable” or “unrecoverable”. Demarcation between the two is explained subsequently. Following are the sequence of steps performed as each UART byte is being received:

1. There is a 6 byte UART message input buffer into which the UART bytes from the UART Rx register are written as they are received. A 3-bit UART message counter is incremented on the reception of each byte in UART message packet i.e. each UART byte received between idle time counter indications .
2. All the subsequent UART bytes in the UART packet are stored internally till any of the following events occurs :
 - UART Timeout i.e. bus idle time is detected.
 - Framing Error from the UART.
 - Number of UART bytes that are received as reported through the UART message counter described in “⁽¹⁾” above exceed 6.
 - STOP mode request mode is asserted. Please see [Section 54.1.4.1, Stop mode](#) for further details.
3. As soon as any of the events described in ⁽²⁾ above occur, it is assumed that a complete UART packet has been received. The XCRC is then calculated immediately on the received UART packet. Please see [Section 54.4.2.5, PSI5-S XCRC handling](#) for details about XCRC. If there is an XCRC error the message is assumed to be “unrecoverable” and it is sent to MBOX₀,LOC₁.The timestamp of the message is captured as is described in [Section 54.4.4, TimeStamp](#) .The watchdog error is captured

as is described in [Section 54.4.5, PSI5-S Watchdog Operation](#).

4. If there is NO XCRC error, then the CID and FID are extracted from the header byte, and the configured length of the message is determined from the configuration in PSI5 Channel n Message Configuration Register A. Following steps are performed .
 - The FID and the CID from the header byte is transferred directly to the PSI5 Frame status word.
 - The transceiver error flags (E0,E1) are transferred directly to the PSI5 Frame status word
 - The message is then analysed to see whether it is “LEGAL” or “ILLEGAL”.
 - Following are the conditions for a “LEGAL” message :

When the CID and the FID of the received message are both “0s” and PS_MSGA_CH0[F0_byte] is NOT “0” OR

When the data of an Asynchronous Enabled channel is received with FID = 0 and PS_MSGA_CHCID[F0_byte] is NOT “0” OR

When the data of a Synchronous Enabled channel is received with legal FIDs(0 to 5) and if the recieved FID is such that the corresponding PS_MSGA_CHCID[F_FID_byte] is NOT 0
 - Any recieved message that does not satisfy the above condition is considered as “ILLEGAL”. Illegal Messages are also treated as “unrecoverable” messages and are sent to MBOX₀,LOC₁.

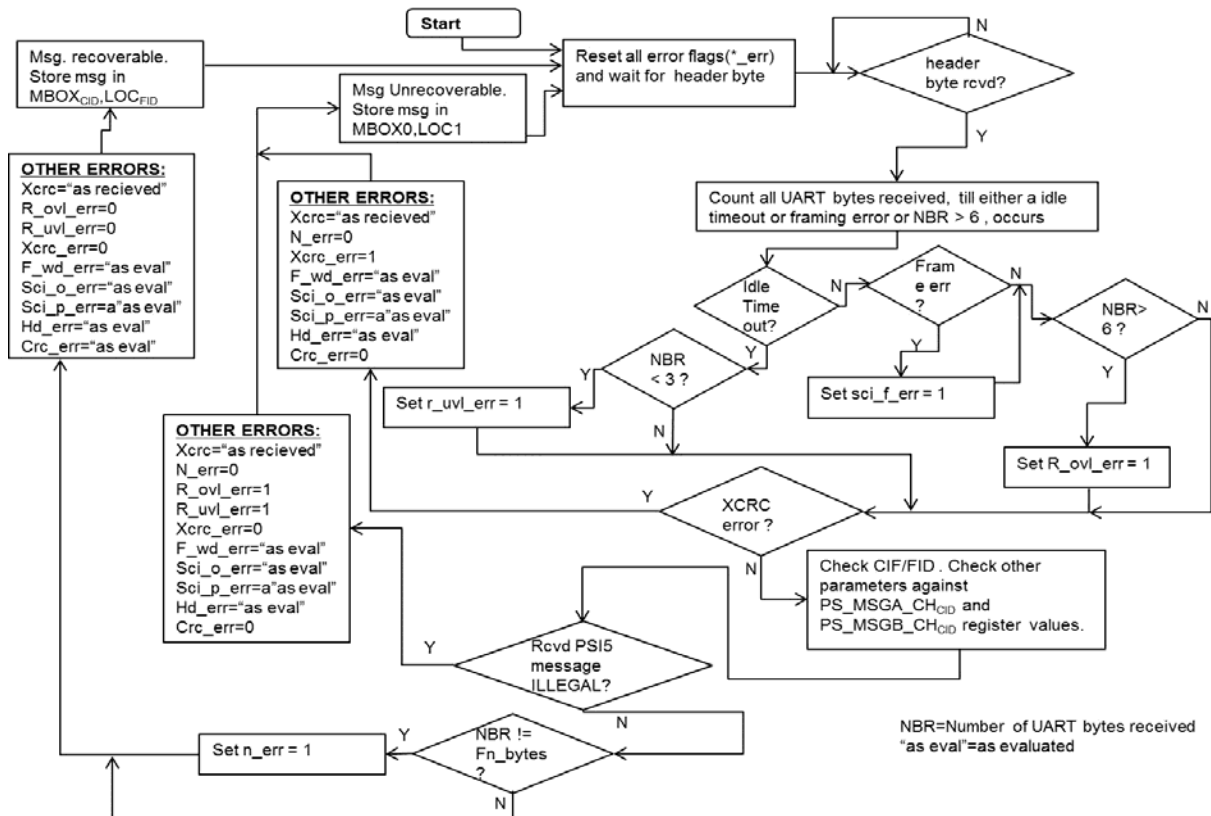
Example can be a message with CID as 0 but FID as 1(illegal because Channel0 is asynchronous and can only have FID equal to 0 messages) or a message recieved for an enabled synchronous channel but having an FID equal to 7(illegal because FID as 7 is not a legal Frame ID) etc.
 - The timestamp of a Legal/Illegal message is captured as is described in [Section 54.4.4, TimeStamp](#) . The watchdog error is captured as is described in [Section 54.4.5, PSI5-S Watchdog Operation](#).
 - The number of UART bytes received in the UART message packet is compared with the number specified in Message Configuration Register A for the channel indicated by CID and FID in the header byte. If the number of bytes indicated by the message counter does not match PS_MSGA_CHCID[F_FID_byte] field , then the status bit N_Err within the PSI5 Frame status word is set. The above “number of bytes recieved” check is not carried out for an “illegal message”.
5. The PSI5 message payload is checked using the PSI5 CRC or parity as per the configuration. Whether the CRC or parity check is correct or not, the PSI5 message payload is written to the PSI5 Sensor Data word and the CRC or parity is written to the PSI5 Frame Status word. Note that the messages which have CID and FID both as “0”, are not checked for CRC or parity error . The CRC/Parity error status bits remain “0” for such messages.This above check is NOT done for “illegal messages”.

54.4.2.2 PSI5-S “Recoverable and Unrecoverable” Messages

Based on [Section 54.4.2.1, PSI5-S Message extraction](#), a message is considered “unrecoverable”, if the XCRC is incorrect or it is an “illegal message”, else the message is considered “recoverable”. “Unrecoverable” message indicates that an error has been detected during the assembling phase of the UART packet i.e., during the formation of the UART packet from the individual UART bytes . “Unrecoverable” messages are always stored in PSI5 mailbox MBOX₀,LOC₁ in system RAM. “Recoverable” messages are always stored in the Mailbox location pointed out by the CID n and the corresponding FID fields.

[Figure 1247](#) describes the treatment of the “Recoverable” and “Unrecoverable” messages. It also details the setting of the various error flags. Note that a stop mode entry (please see [Section 54.1.4.1, Stop mode](#)) also causes the flushing of the internal PSI5 message buffer and this event is treated like a new message completion. This is however not shown in [Figure 1247](#).

Figure 1247. PSI5-S message handling and Error Flags



Once the message has been correctly deciphered as “Recoverable” or “Unrecoverable” and the various error flags have been set then the Message is finally transferred to the MRU buf2. Once the corresponding message has been read from the MRU buf2 the bits in the “PS_MBOX_SR_CH[n]” register are set appropriately. This aids the application to correctly read the concerned message from the Mailbox. [Section 54.4.2.3, PSI5-S Message Storage and Mailbox Transfer](#) describes the PSI5 Message Storage and detection in detail.

54.4.2.3 PSI5-S Message Storage and Mailbox Transfer

Once the message has been correctly deciphered as “recoverable” or “unrecoverable” and the various status bits are set, it is then transferred to the MRU output buffer. The MRU output buffer is a 2 stage buffer differentiated as MRU Buf1 and MRU Buf2. The MRU buf1 is composed of 3, 32 bit registers while the MRU buf2 is composed of 4, 32bit registers. *Figure 1248* shows the structure of MRU buf1. It is composed of three internal registers. These registers are not in the addressable space of the IP. When the internal state machine has correctly decoded the PSI5 message, it is transferred to MRU buffer1. It serves to hold the PSI5 message until it is transferred to MRU buf2.

Figure 1248. MRU buf1 structure

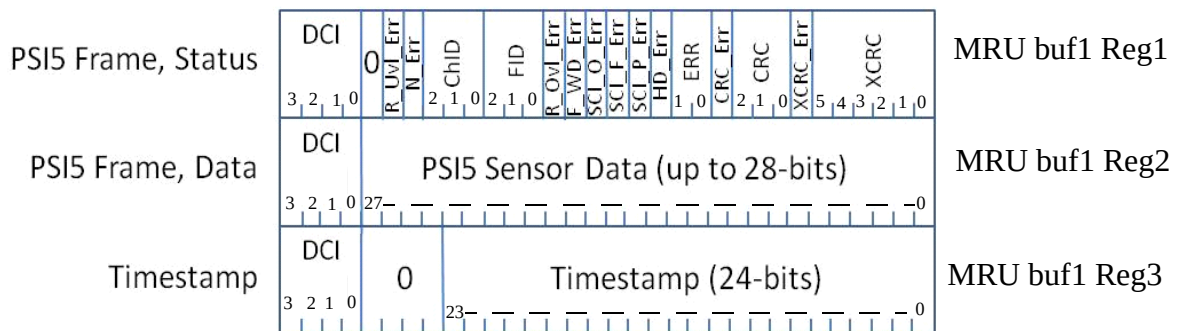
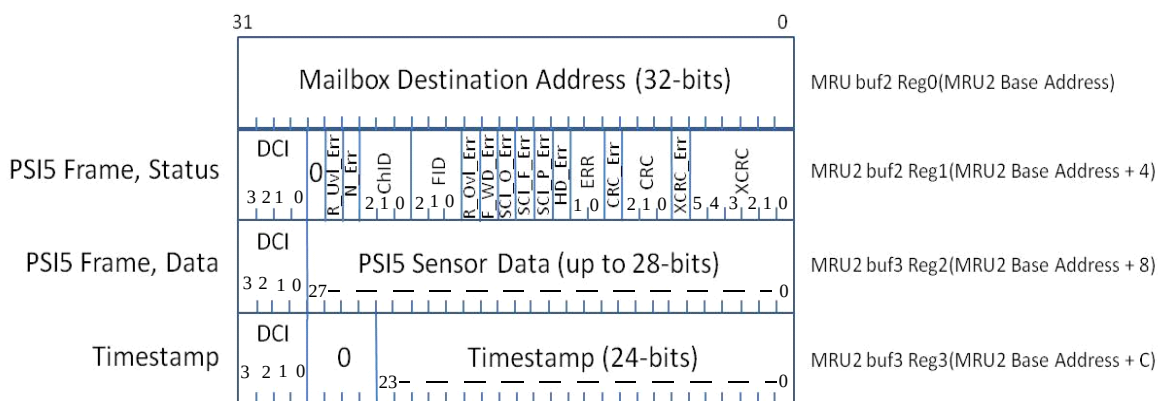


Figure 1249 shows the structure of MRU buf2. MRU buf2 has one extra location as compared to MRU buf1. This extra location (MRU buf2 Reg0) contains the address of the Mailbox location, where the message contained in the rest of the three locations of MRU buffer2 is to be placed. This specific MRU buf2 structure aids the DMA to perform linked scatter-gather data transfer from the IP. The extra location in the MRU buf2 is automatically updated by the hardware and provides the destination address for the message. This destination is the Mailbox Address at which the message (stored in MRU buf2 (Reg1-Reg3)) is to be transferred by the DMA. This destination address is calculated by the IP based on ChID in the message header (or ChID0 in the case of an unrecoverable message), the FID in the message header and Base address of the mailbox in system RAM (provided by CH_BASE_ADDR) register.

Figure 1249. MRU buf2 structure



MRU buf2 is in the addressable space of the IP and is read by the DMA when the corresponding DMA request is asserted. The PSI5 message is transferred from the MRU buf1 to the MRU buf2 only when the MRU buf2 has been read completely by the DMA, i.e., all the 4 locations of the MRU buffer2 have been read by the DMA, for the corresponding PSI5 message.

Following are the sequence of operations that are performed, before the message in MRU buf2, gets ready to be accessed by the DMA:

1. The IP decodes the message as detailed in [Section 54.4.2.1, PSI5-S Message extraction](#) and [Section 54.4.2.2, PSI5-S “Recoverable and Unrecoverable” Messages](#).
2. The corresponding message is available in MRU buf1 along with the various status bits latched internally in the IP. Externally, no interrupt is generated and neither are any status bit updated, when the message is transferred to MRU buf1. When the message is transferred to MRU buf1 a 3 bit DCI (data consistency indicator) is added. The DCI is a 3-bit counter that is counted on a reception of each new message in MRU buf1.
3. The IP checks whether MRU buf2 is empty. If it is empty, then the MRU buf1 “Reg1 to Reg3” contents are transferred to the MRU buf2 “Reg1 to Reg3”. The corresponding interrupts (except the Message Read from MRU) are also asserted based on the Interrupt Enable settings. Based on the PSI_CH_BASE_ADDR, the CID, FID values and whether an “unrecoverable message” has been received, the IP calculates the contents of MRU buf2 Reg0 register.
 - If CID == 0 then the “MRU buf2 Reg0” contains the Mailbox Location corresponding to MBOX0,LOC0.
 - If “unrecoverable message” is received then “MRU buf2 Reg0” contains the Mailbox Location corresponding to MBOX0,LOC1.
 - In all other cases “MRU buf2 Reg0” contains the Mailbox location corresponding to the MBOX_{CID},LOC_{FID}.
4. Once the calculations in ⁽³⁾, are done the IP asserts the DMA request for Access1. In DMA Access 1, the DMA reads the first location (MRU buf2 Reg0) of the MRU buf2. This single 32-bit access writes the content of MRU buf2 Reg0 into the destination address for the DMA channel performing the second access (Access 2). Access 2 is initiated by the DMA once the Access1 transfer is complete. Access 2 is composed of 3,32-bit accesses of MRU buf2 and it writes the reconstructed PSI5 message along with its status and timestamp to the mailbox, starting at the destination address provided by Access 1. In Access2 the DMA reads the MRU buf2 Reg1,Reg2 and Reg3 in that order.
5. The application can read the PS_MBOX_SR_CH[n] register to get an indication to which mailbox location (or locations) the DMA would transfer the message. Note that “DMA transfer complete” bit/interrupt of the DMA indicates to the application, that the message read by the DMA from the IP has actually been written to the Mailbox.
6. The application should clear the corresponding “Fn_read” bits in the PS_MBOX_SR_CH[n] register once the application has read the corresponding message from the Mailbox. If a PSI5 message gets written to a Mailbox location, with “Fn_read” bit still set at the time at which the DMA has started the Access1, then the corresponding “Fn_OV” bits are set in the PS_MBOX_SR_CH[n] register.
7. Reference to ⁽³⁾ above, if the MRU buf2 has a pending message and the MRU buf1 also has a message that is ready to be transferred to the MRU buf2 and then a new message gets ready to be written to MRU buf1, then the previous message in MRU buf1 gets overwritten by the new message. The MRU overwrite interrupt is also generated with the setting of the corresponding register bit in the PS_GLSR register. PS_GLSR register contains the CID and the FID of the message that got overwritten by the new message in MRU buf1.
8. When writing an unrecoverable message to the PSI5 mailbox CID0, with the XCRC_ERR bit set, the timestamp that will be used will be the one which is captured at the time XCRC_ERR bit is set. When the unrecoverable message is due to an “illegal message”, then the timestamp used would be the one which is captured on the

reception of any message i.e., always captured at the end of the header byte of every message.

9. In Asynchronous mode, FID will always be configured to be “0” and the module will use the PSI5 Mailbox for the channel (identified by CID in header) as a ring buffer, incrementing the destination address for each message received and saving up to the last six messages received. The bits of the PS_MBOX_SR_CH[n] register will still indicate the status of each location of the Ring Buffer. The software has to still clear the six bits of this register when it reads the Mailbox location else the overwrite bits would be set in the PS_MBOX_SR_CH[n] register. The message identification mechanism allows software to identify the latest message received

54.4.2.4 PSI5-S Data Payload CRC Recalculation

Error detection is realized by a single bit even parity (recommended for 10 or fewer bits) or a 3-bit CRC (recommended for long data words). The transmission error detection is selectable (by frame configuration register) as:

- 1-bit even parity
- 3-bit CRC

The generator polynomial of the CRC is $g(x)=1+x+x^3$ with a binary CRC initialization value of 111. The transmitter extends the data bits by three 0s (as MSBs). This augmented data word is fed (LSB first) into the shift registers of the CRC check. Start bits are ignored in this check. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the buffer registers contain the CRC checksum. These three check bits are transmitted in reverse order (MSB first: C2, C1, C0).

Figure 1250. CRC calculation scheme

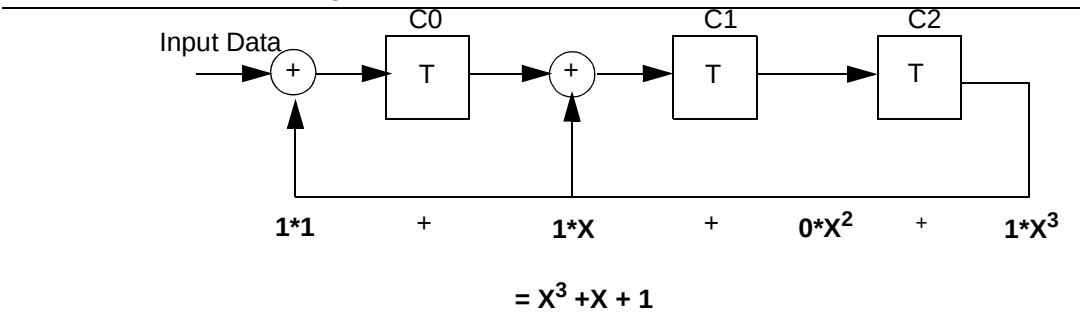


Table 1224. CRC Calculation (at transmitter side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	0	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	1	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	1	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	1

Table 1225. Transmitted and Received Sensor Data

Transmitted Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0
Received Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

Table 1226. CRC Re-Calculation (At Receiver Side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	1	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	0	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	0

Syndrome = 000 → Transmission successful

The correct transmission is checked by the PSI5 receiver by comparing the syndrome (content of the shift register after reception of both data bits and CRC bits) with the zero vector. For error free reception, all shift registers have to be zero.

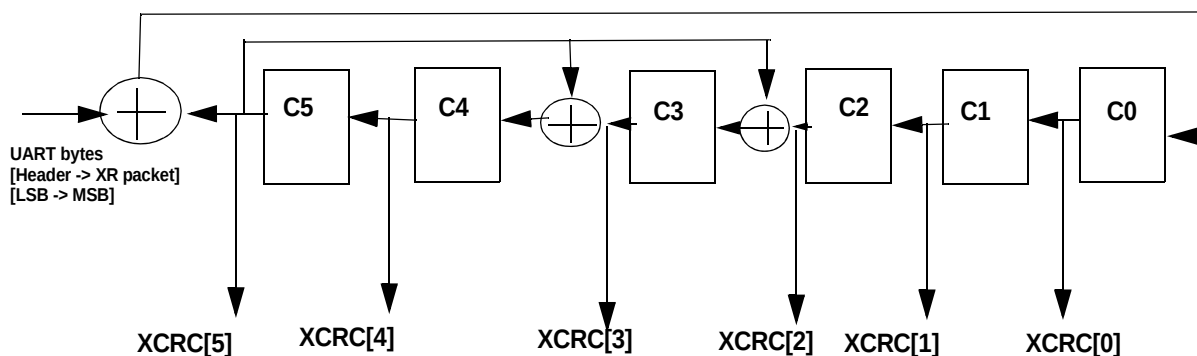
54.4.2.5 PSI5-S XCRC handling

The XCRC calculation is triggered on any of the following events(which also indicate the reception of a new UART packet):

- UART idle timeout
- number of UART bytes exceed 6
- UART Framing error
- STOP mode assertion. Please see [Section 54.1.4.1, Stop mode](#).

When any of the above events occur , the internal XCRC calculator becomes active. Each UART byte that is received, passes through the XCRC calculator on a bit by bit basis, starting from the header byte to the expected XCRC byte. The XCRC is assumed to be the last 6 bits of the last UART byte that is recieved before the occurrence of any of the above events. [Figure 1251](#) shows the XCRC calculation algorithm.

Figure 1251. XCRC calculation



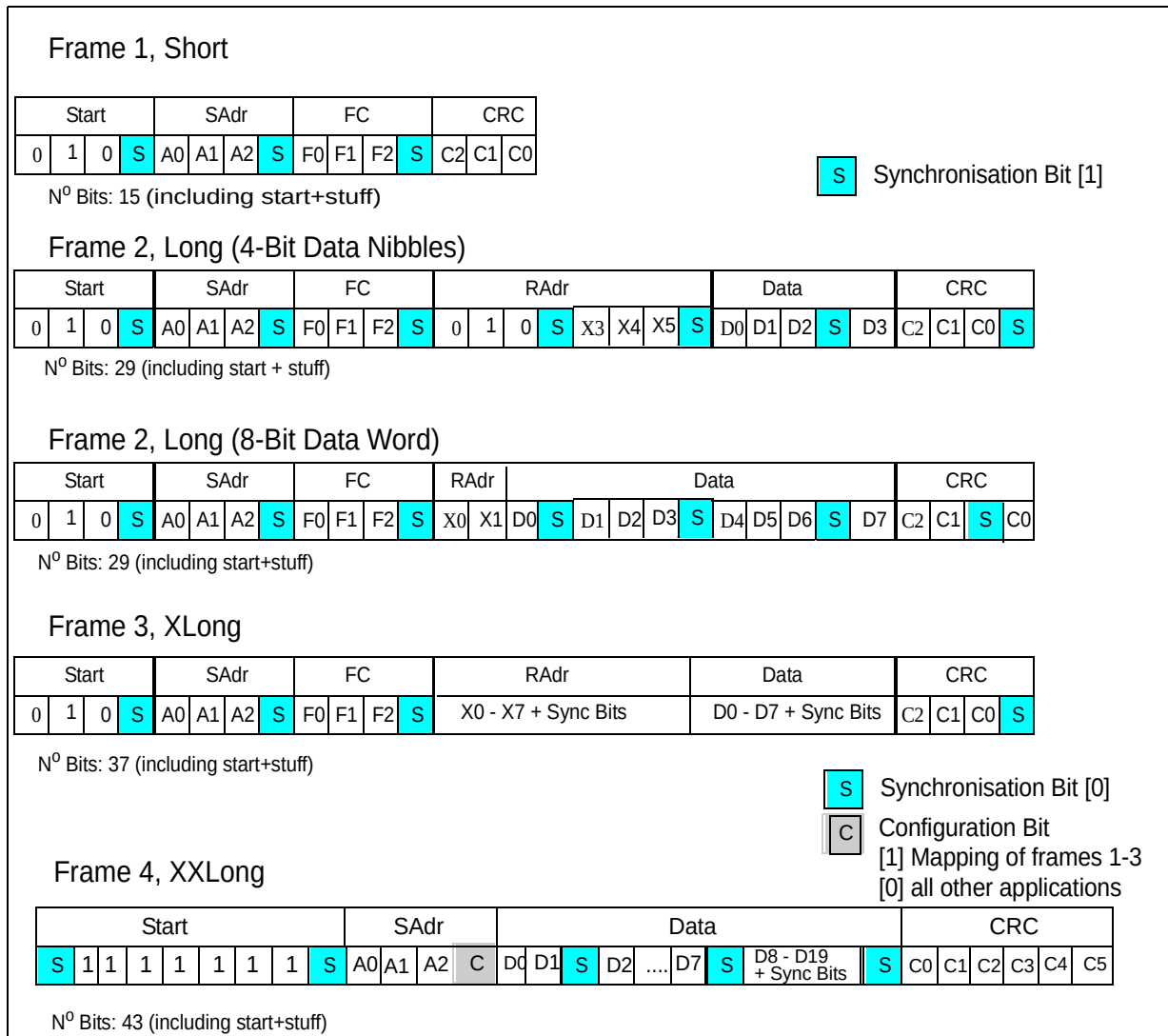
The UART bytes are shifted starting from the received header to data packets to the XCRC packet. Each of these UART bytes is shifted LSB first into this XCRC calculator. Once all the bytes as denoted by the Fn bytes have been received and the Fnth packet (XCRC) has been shifted completely through this XCRC calculator and if XCRC[5:0] is equal to "0" then it denotes that the XCRC check has passed else there is an XCRC error and the XCRC_ERR bit is set.

The CRC polynomial used is " $x^6+x^4+x^3+1$ " with a binary initialization value of XCRC[5:0] = "010101".

54.4.3 ECU-to-sensor communication

The IP send the commands for the ECU to Sensor communication in terms of UART bytes. It expects the external transceiver to correctly interpret the commands and then generate the appropriate logic 0 or logic 1 on the PSI5 bus. Following are the various PSI5-S ECU to Sensor commands supported by the IP, as shown in [Figure 1252](#).

Figure 1252. Standard Frame Formats Supported



- Short:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- **"010",S, "A0,A1,A2",S, "F0,F1,F2",S, "C2,C1,C0"** where "010" is the start sequence "S" is the stuff bit(1),A0-A2 is the sensor address,F0-F2 are the functional codes and "C2-C0" is the CRC. Please refer [Figure 1252](#) for the frame structure.
 - The CRC polynomial used is $x^3 + x + 1$ with a binary initialization value of "111(C2:C0)".
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,"000"] with "A0" being given first and "0" as the last. When the

last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.

- When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]....PS_DDSR[5],000) with PS_DDSR[0] being fed first in the CRC calculator.
- **Long(4bit Data Nibble):** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- “010”,S, “A0,A1,A2”,S, “F0,F1,F2”,S, “X0,X1,X2”,S, “X3,X4,X5”,S, “D0,D1,D2”,S, “D3,C2,C1”,S, “C0” where “010” is the start sequence “S” is the stuff bit(1),A0-A2 is the sensor address,F0-F2 are the functional codes,”X0-X5” is the Range Address, “D0-D3” is the data and “C2-C0” is the CRC. Please refer [Figure 1252](#) for the frame structure.
 - The CRC polynomial used is $x^3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0..X5,D0...D3,”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
 - When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]....PS_DDSR[15],000) with PS_DDSR[0] being fed first in the CRC calculator.
- **Long(8bit Data Word):** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - “010”,S, “A0,A1,A2”,S, “F0,F1,F2”,S, “X0,X1,D0”,S, “D1,D2,D3”,S, “D4,D4,D6”,S, “D7,C2,C1”,S, “C0” where “010” is the start sequence “S” is the stuff bit(1),A0-A2 is the sensor address,F0-F2 are the functional codes,”X0-X1” is the Range Address, “D0-D7” is the data and “C2-C0” is the CRC.Please refer [Figure 1252](#) for the frame structure.
 - The CRC polynomial used is $x^3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0..X1,D0...D7,”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
 - When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]....PS_DDSR[15],000) with PS_DDSR[0] being fed first in the CRC calculator.
- **X-Long:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - “010”,S, “A0,A1,A2”,S, “F0,F1,F2”,S, “X0,X1,X2”,S, “X3,X4,X5”,S, “X6,X7,D0”,S, “D1,D2,D3”,S, “D4,D5,D6”,S, “D7,C2,C1”,S, “C0” where “010” is the start sequence “S” is the stuff bit(1),A0-A2 is the sensor address,F0-F2 are the functional codes,”X0-X5” is the Range Address, “D0-D3” is the data and “C2-C0” is the CRC. Please refer [Figure 1252](#) for the

frame structure.

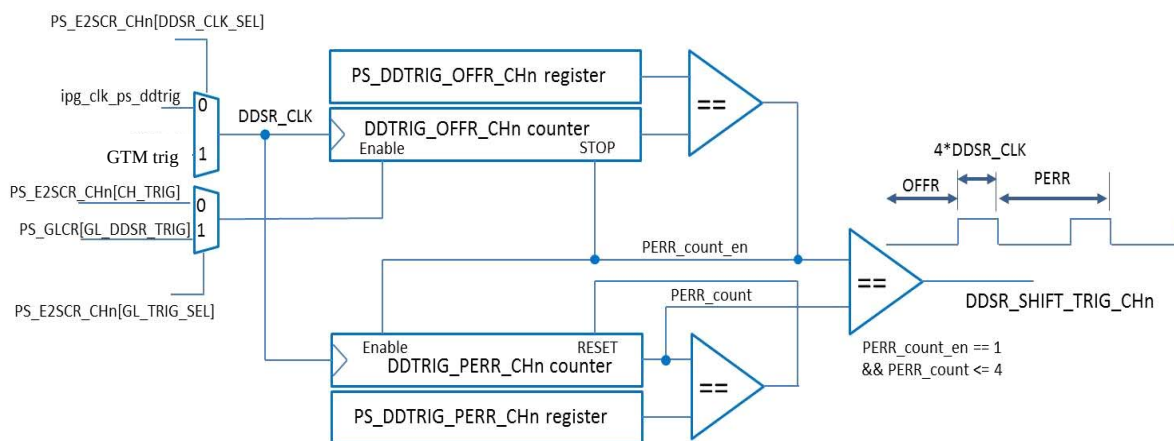
- The CRC polynomial used is $x^3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0...X7,D0...D7,”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
 - When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]....PS_DDSR[21],000) with PS_DDSR[0] being fed first in the CRC calculator.
- **XX-Long:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C5 comes Last) Data Stream - “01111110”, “A0,A1,A2”, “C”, “D0,D1”,S, “D2,D3,D4” D5,D6,D7, S, “D8,D9,D10,D11,D12,D13”, S, “D14,D15,D16,D17,D18,D19”,S, “C0,C1,C2,C3,C4,C5” where “01111110” is the start sequence “S” is the stuff bit(0),A0-A2 is the sensor address, “C” is the configuration bit, “D0-D19” is the data and “C0-C5” is the CRC. Please refer [Figure 1252](#) for the frame structure.
 - The CRC polynomial used is $x^6 + x^4 + x^3 + 1$ with a binary CRC initialization value 010101(C5:C0).
 - The transmitter extends the data bits by six zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,D0..D19,”000000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
 - When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]....PS_DDSR[23],000000) with PS_DDSR[0] being fed first in the CRC calculator.

54.4.3.1 PSI5-S ECU to Sensor Command Handling

The ECU-to-Sensor commands can be written to the PSI5 IP in two ways:

1. Direct writing to the UART Tx Buffer : This can be used by the software to directly write the transceiver/diagnostic commands to the UART Tx buffer . In case there is a simultaneous UART access also by the PSI5-S core then the priority is given to the system(software) . The pending command from the PSI5-S core is then buffered internally and is transmitted when the system is no longer accessing the UART core. The system can write to the UART Tx buffer when the PS_UARTSR[DTF] flag goes to “1”. Setting of this bit indicates that the UART Tx buffer is ready for accepting a new byte.Interrupt can also be generated by setting the corresponding interrupt enable bit to “1” .
2. Writing to the DDSR (Downstream Data Shift Register): In this case the application writes the basic command to the DDSR_L and the IP will add the start bits/start condition/CRC/stuff bits before sending this command to the UART Tx FIFO. Following are the details:
 - There is one DDSR per channel. The default value of DDSR is dependent on the PS_E2SCR[CMD_TYPE] value. DDSR is composed of two sub registers DDSR_H (Higher) and DDSR_L (Lower). The application can only write to DDSR_L. The number of bits that can be written in the DDSR_L is dependent on

- the value of PS_E2SCR[CMD_TYPE] bit values. Note that Channel0 does not have any DDSR, as it does not support any ECU-to-Sensor communication.
- The “shift trigger” of the DDSR is governed by the values programmed in the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n] register and the frequency of ipg_clk_ps_ddtrig clock .
 - Once the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n] are correctly programmed and each channel is programmed to respond to the selected shift trigger (local trigger or global trigger is decided via the bit PS_E2SCR[GL_TRIG_SEL]) the software has to write to the Global DDSR trigger bit, PS_GLCR[GL_DDSR_TRIG] or the PS_E2CR[CH_TRIG] bit, so as to synchronously start the DDSR triggers of each channel or locally start the DDSR shift triggers of each channel. The DDSR trigger logic gets the trigger signal immediately but it responds after a delay as programmed in the PS_DDTRIG_OFFR_CH[n] register. The shift trigger period is governed by the value programmed in the PS_DDTRIG_PERR_CH[n] register. Please see [Figure 1253](#) for the structure of the logic that generates the internal DDSR shift triggers.

Figure 1253. DDSR internal shift trigger generator

Note : This structure is replicated for each channel except Channel0, in which this structure is not present.

- The number of bits of DDSR_L that can be written by the application, is dependent on the value of PS_E2SCR[CMD_TYPE] value.
- When PS_E2SSR[DDSR_RDY] == 1, then it indicates that the DDSR is empty and is ready to accept a new command. If the PS_E2SCR[DDSR_RDY_IE] == 1 then this results in the generation of interrupt on the “ipi_ps_e2s_ch[n]” line.
- Attempting to write to DDSR while PS_E2SSR[DDSR_RDY] == 0 results in the rejection of the new command and setting of the PS_E2SSR[CMD_NWRT] bit. If the PS_E2SCR[CMDTR_NWRT_IE] == 1 then this results in the generation of interrupt on the “ipi_ps_e2s_ch[n]” line.
- The application then writes the command to the DDSR and writes a “0” to the PS_E2SCR[DDSR_RDY]. This indicates to the internal logic that the command is

now ready for further processing. The clock that is used for shifting the contents of the DDSR can be configured through the PS_E2SCR[DDSR_CLK_SEL] register.

- Depending on the value of PS_E2SCR[CMD_TYPE], the internal logic adds the start bits/start condition/CRC/bit stuffing and updates the DDSR with the complete command, ready for transmission.
- Once this happens the PS_E2SSR[CMDPR_BZY] becomes “1”. If the PS_E2SCR[CMDPR_BZY_IE] == 1 then this results in the generation of interrupt on the “ipi_ps_e2s_ch[n]” line. If PS_E2SCR[CMDTR_SW_CTRL] == 0 then the application can check the final value of the command in DDSR, that would actually be transmitted. Once the checking is complete it can put the PS_E2SCR[CMDTR_SW_CTRL] to “1”, to start the actual transmission. By default PS_E2SCR[CMDTR_SW_CTRL] is set to “1”. Note that the start condition is not a part of the DDSR and its transmission is dependent on the value of the PS_E2SCR[CMD_TYPE] and is automatically taken care by the IP.
- Each bit of the command in DDSR, if it is “1”, results in PS_E2SCR[CMD] being transferred to the UARTTX buffer or if it is “0” results in PS_E2SCR[ACMD] being transmitted to the UART-TX buffer. The “CMD” and the “ACMD” governs as to how the transceiver has to respond when it receives these values. These will specify to the transceiver whether it should send a short or long SYNC pulse on the PSI5 bus when the application wishes to write to the sensors
- Once the complete command has been transmitted the PS_E2SSR[DDSR_RDY] goes to “1” while PS_E2SSR[CMDPR_BZY] goes to “0”.
- Once the PS_E2SSR[DDSR_RDY] goes to “1” then default “1s” or default “0s” are transmitted, depending on the value of PS_E2SCR[CMD_TYPE].
- When PS_E2SSR[DDSR_RDY] == 1, then it is possible to abort the current transmission by doing a “W1C” of the PS_E2SCR[DDSR_CLR] bit. As soon as this is done the DDSR is reset to “0”, PS_E2SSR[DDSR_RDY] goes to “1” while PS_E2SSR[CMDPR_BZY] goes to “0”. Once the PS_E2SSR[DDSR_RDY] goes to “1” then default “1s” or default “0s” are transmitted, depending on the value of PS_E2SCR[CMD_TYPE]. Note that doing this “W1C” does not have any effect on the CMD/ACMD transmission that is already under progress in the UART.

Figure 1254. PS_DDSR Sensor Command (Ex: XXLong Frame)

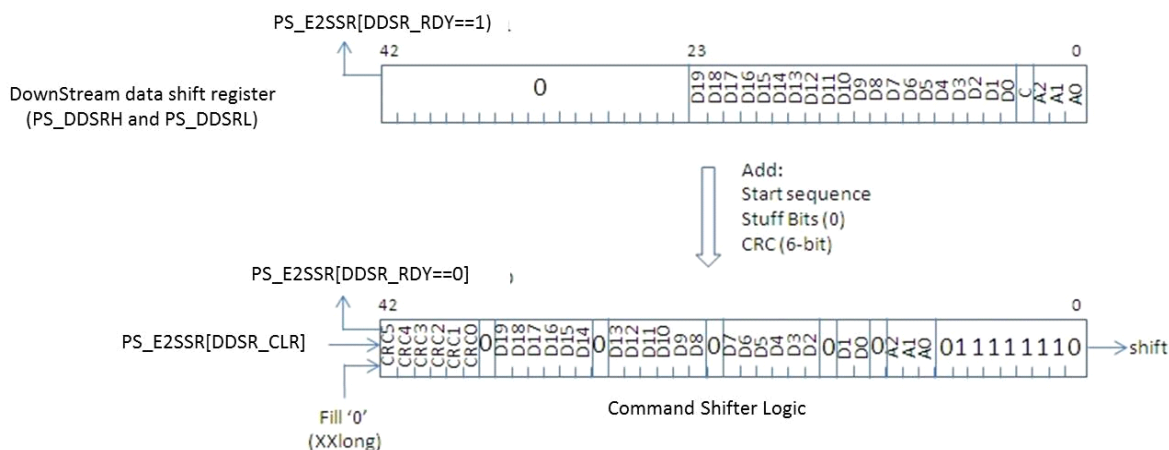


Figure 1255. PS_DD SR Sensor Command (Ex: PSI5 LONG format)

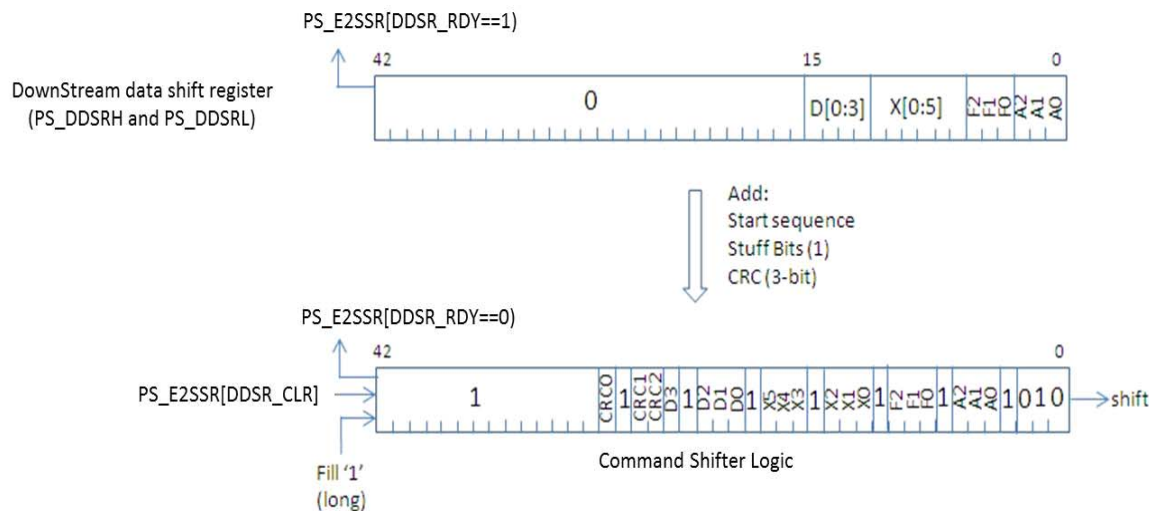
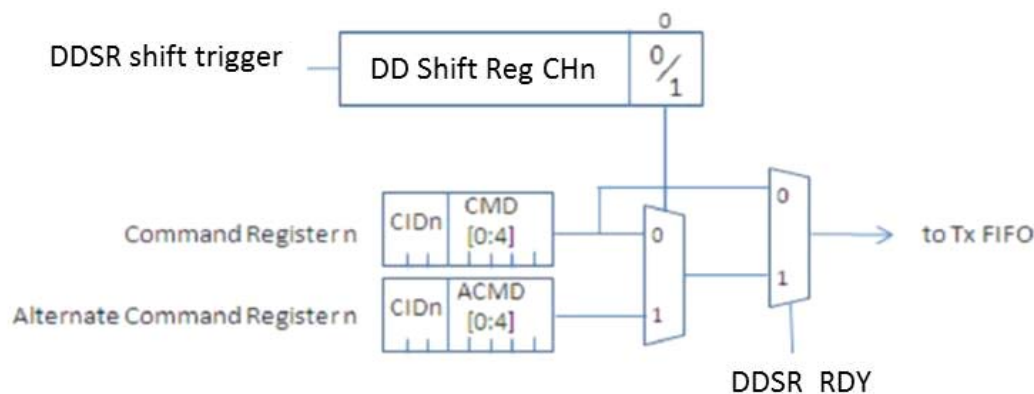


Figure 1256. CMD and ACMD



- The logic for processing the DDSR contents is shared among all the 7 channels. It has to be ensured by the software to correctly program the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n] register contents so that the none of the bits in the command get missed. Also since ⁽¹⁾ is possible, it is up to the application to ensure that the UART TX_BUFFER does not remain busy to the extent that some particular DDSR bit does not get the chance to be shifted out before it gets overwritten by some new DDSR bit.
- The approximate maximum time, beginning from when PS_E2SCR[DDSR_RDY] is made "0", to the time when the command is available for transmission is less than 1us, provided no other channel has an active command processing ongoing. If commands are written to multiple DDSR registers one after the other, then the time after which the particular channel will again get an access will

increase. Note that the irrespective of the sequence in which the PS_E2SCR[DDSR_RDY] of the channels are cleared, the channels are always searched in a round robin fashion for the status of the respective DDSR_RDY bits. When the internal sequencer find the respective DDSR_RDY as "0" it starts the processing of the corresponding DDSR. The sequencer always goes as Channel1->Channel2->Channel3...Channel7->Channel1...

- The logic that shifts each bit of the DDSRs of each channel is shared among all the channels. When the shift trigger of a particular DDSR is received and if the UART TX Buffer is empty then the particular command bit is transmitted to the UART TX by the appropriate CMD/ACMD. It is possible that at one point of time , CMD/ACMD of more than one channel are pending to be transmitted to the UART TX . In this case the channel with a higher channel number will be given a priority . Thus if channel2 and channel1 , both have a pending CMD/ACMD shift, then channel2 would be the first to get an access .Each pending channel would queue up for transmission . It has to be ensured by the software that the internal shift triggers generated through the PS_DDTRIG_PERR_CHn and PS_DDTRIG_OFFR_CHn registers are correctly spaced so that all the channels get an access to transmit their DDSR bits. If a command bit that is desired to be shifted encounters more than one shift triggers (because it did not get shifted on the previous shift trigger) then there is no indication for this loss of synchronisation . In general , if the if PS_DDTRIG_PERR is programmed to be greater than 500us(which is a practical PSI5 sync pulse period) then it would be ensured that the bits are correctly transmitted even if all the channels simultaneously transmit their command bits.

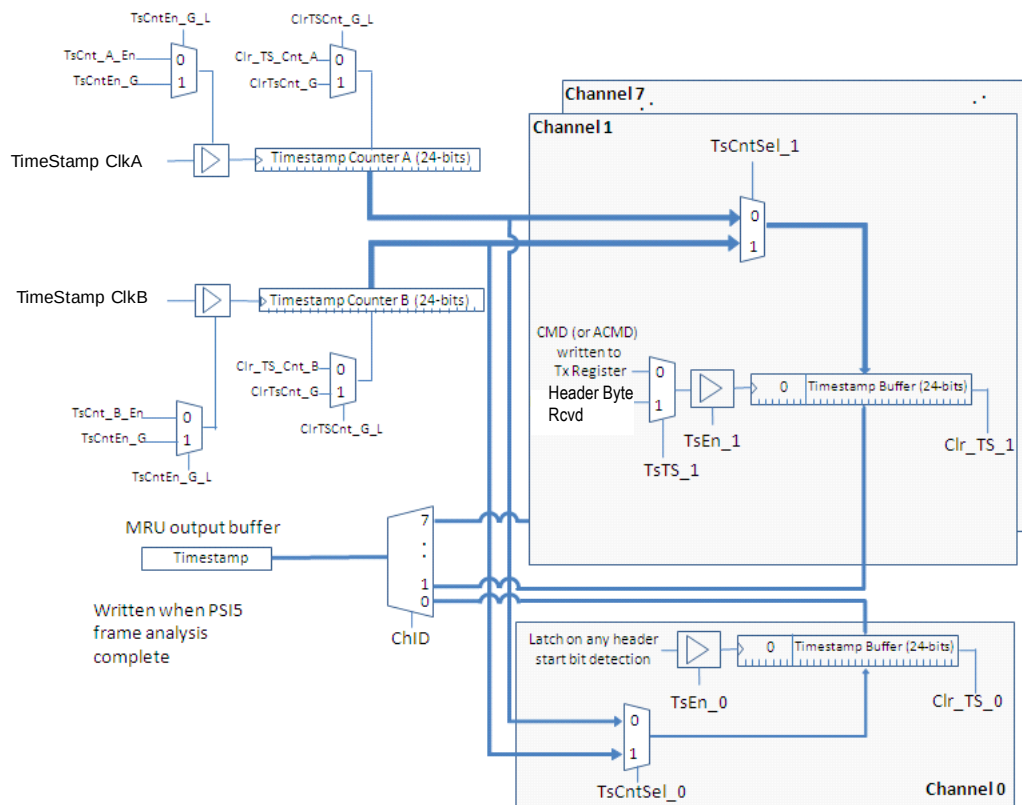
54.4.4 TimeStamp

The IP provides for two global 24-bit timestamp counters. Each can be clocked by either the module clock or the GTM trigger. The two timestamp counters are detailed as TimeStamp Counter A and B. Please see [Figure 1257](#). Following are the details:

1. Each channel has an internal TimeStamp buffer that can be configured to capture either the TS counter A or B when it encounters the "capture event". Provided there is no XCRC error in the message, then this Timestamp buffer is appended to the message when it is written to MRU buf1 . When there is an XCRC error in the message, then Channel0 timestamp is appended. Please see point number ⁽⁷⁾ below.
2. The "captured event" for the timestamp buffer of each channel can either be when a "a valid message header is detected" or when "a Command (or Alternate Command) is moved from the UART Tx Buffer to the UART Tx shift register" corresponding to the channel for which this command is being shifted to the UART .
3. The TimeStamp Buffer can be enabled or disabled through the PS_MSGA_CHn[TSBUF_EN] bit . When the TimeStamp buffer is disabled then it no longer responds to "capture events" and always contains the value "0".
4. It is possible to reset each TimeStamp buffer by using the PS_MSGA_CHn[TSBUF_CLR] bit . As soon as this bit is written to "1" the TimeStamp buffer is reset to all "0s" and it remains in this state till it encounters a "capture event".
5. In the case when the reception of message is chosen as the trigger, the time stamp will be captured immediately after the header byte of the UART message packet has been received.
6. In the case when the move of CMD or ACMD from the internal UART Tx Buffer to the UART Tx shift register in the UART module is chosen as the trigger, the timestamp for the intended channel (indicated in the CIDn fields of CMD or ACMD) is valid for all

- frames received on a channel between moves of CMD or ACMD from the UART Tx Buffer to the UART Tx Register (i.e., for a PSI5 SYNC period).
7. When a XCRC error is detected in any message then the timestamp that is appended with the message is the Channel0 timestamp which is captured at the time when XCRC_ERR has been detected.
 8. For “illegal messages”, the timestamp value that is captured is the Channel0 Timestamp value, which is captured whenever “a valid message header for that illegal message is detected”. Note that the definition of a “legal” or a “illegal” message applies to a PSI5 message only when the concerned UART packet has no XCRC error.
 9. In addition to the timestamp buffers described in point number ⁽¹⁾ above, there is a separate timestamp capture that occurs in case of a watchdog error. This timestamp is captured in the PS_WDGTSSR[WDGTS_STATUS] bits. The TimeStamp A or B (as programmed in the PS_MSGA_CHn[TIMESTAMP_A_B_SEL]) is captured as soon as the watchdog overrun error occurs for that channel. Please see [Section 54.3.16, PSI5-S Watchdog Error Status and Watchdog Timestamp status register \(PS_WDGTSSR\)](#) for more details.

Figure 1257. TimeStamp Capture



54.4.5 PSI5-S Watchdog Operation

The IP provides a watchdog to keep track of a number of reception parameters. The watchdog counter is a 24 bit down counter. Each channel, except Channel0, contains a watchdog counter. The refresh event for the watchdog is automatically chosen depending on whether the channels mode is synchronous or asynchronous. The watchdog timeout

value can be configured in the “WD_TO” bits in the PS_WD_CFGR_CH[n] register. The refresh event for the watchdog are different in the synchronous and asynchronous modes as mentioned below :

- In Asynchronous mode :

Whether the WD has to be “refreshed” or “not to be refreshed” is analyzed when the XCRC is calculated . In case the XCRC is correct the watchdog is refreshed (It is not STOPPED) , else the WD counter keeps on running normally eventually leading to a watchdog overrun error.

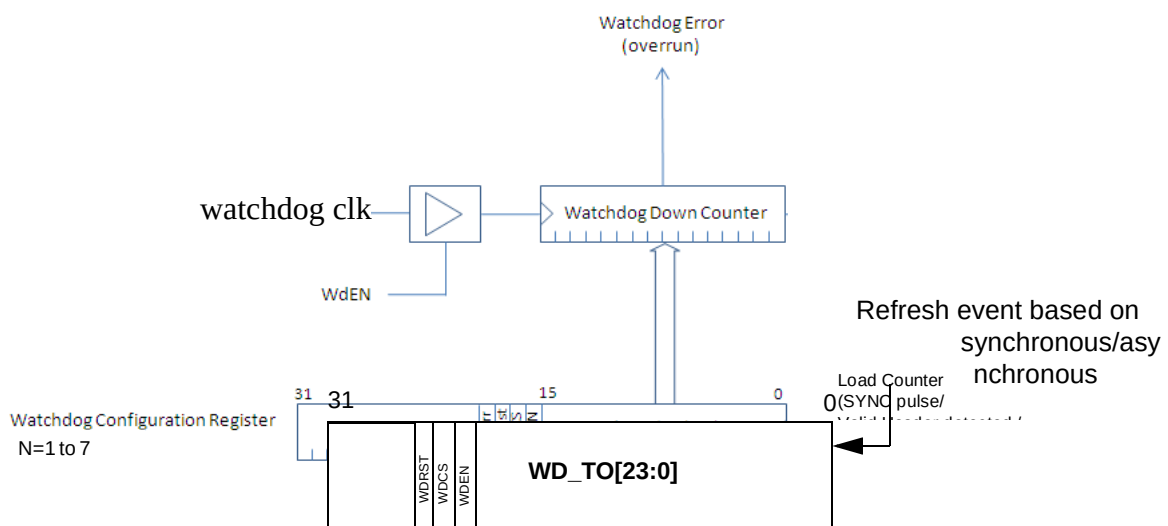
- In Synchronous Mode:

The watchdog is ALWAYS refreshed on the occurrence of the internal sync pulse and keeps on counting thereafter. However, it is stopped when the “Packet_counter” reaches the count of “total number of configured slots”. The “Packet_counter” here is a counter that keeps track of the number of configured(enabled) slots. It is incremented on the reception of each UART packet with a “CID” and “FID” such that PS_MSGA_CH_CID[FID_byte] is not equal to “0” . The packet counter is however incremented ONLY if the XCRC of the received UART packet is correct else it is not incremented. In case the “Packet Counter” does not reach the “expected packet count” within the “WD timeout configured “ , the WD overrun error is generated.

In both the above modes , the watchdog is always refreshed on the occurrence of Watchdog overrun error (F_WD_ERR) or a software reload . Also please refer to [Section 54.3.3.23, PSI5-S channel watchdog configuration register \(PS_WD_CFGR_CHn\) \[n==1..7\]](#)

The watchdog can be started in the NORMAL mode by writing a “1” to the WDEN bit in the PSI5_WD_CFGR_CH[n]. If the watchdog counter is not refreshed by its event and the watchdog counter becomes equal to “0”; the F_WD_ERR flag is set in the PSI5_IRQ_ERR_SR register and the same is also reflected in the PS_MRU_BUF2_REG1 register. An interrupt can also be generated on any of the “ipi_ps_irq[irq_n]” lines depending on the configuration. Note that writing a “0” to WDEN bit disables the watchdog checking mechanism. [Figure 1258](#) shows the watchdog counter.

Figure 1258. Watchdog Counter



As soon as the watchdog overrun error for a particular channel occurs , the status of the watchdog error is immediately latched in the register described in [Section 54.3.3.16, PSI5-S](#)

Watchdog Error Status and Watchdog Timestamp status register (PS_WDGTSSR). This watchdog error status(PS_WDGTSSR[F_WD_ERR_STATUS[CHn]]) is also sent to the MRU registers(as F_WD_ERR) as shown in the PSI5 frame status word (shown in [Figure 1248](#) and [Figure 1249](#)) . However this information is sent in the next upcoming PSI5 message packet that occurs for that particular channel , provided this upcoming PSI5 message has no XCRC error. In case the upcoming UART packet has an XCRC error then the Frame watchdog error field in this PSI5 message packet is the “OR” of the F_WD_ERR status of all the channels i.e. OR of all the fields of PS_WDGTSSR[F_WD_ERR_STATUS] bits.

54.4.6 Interrupt handling

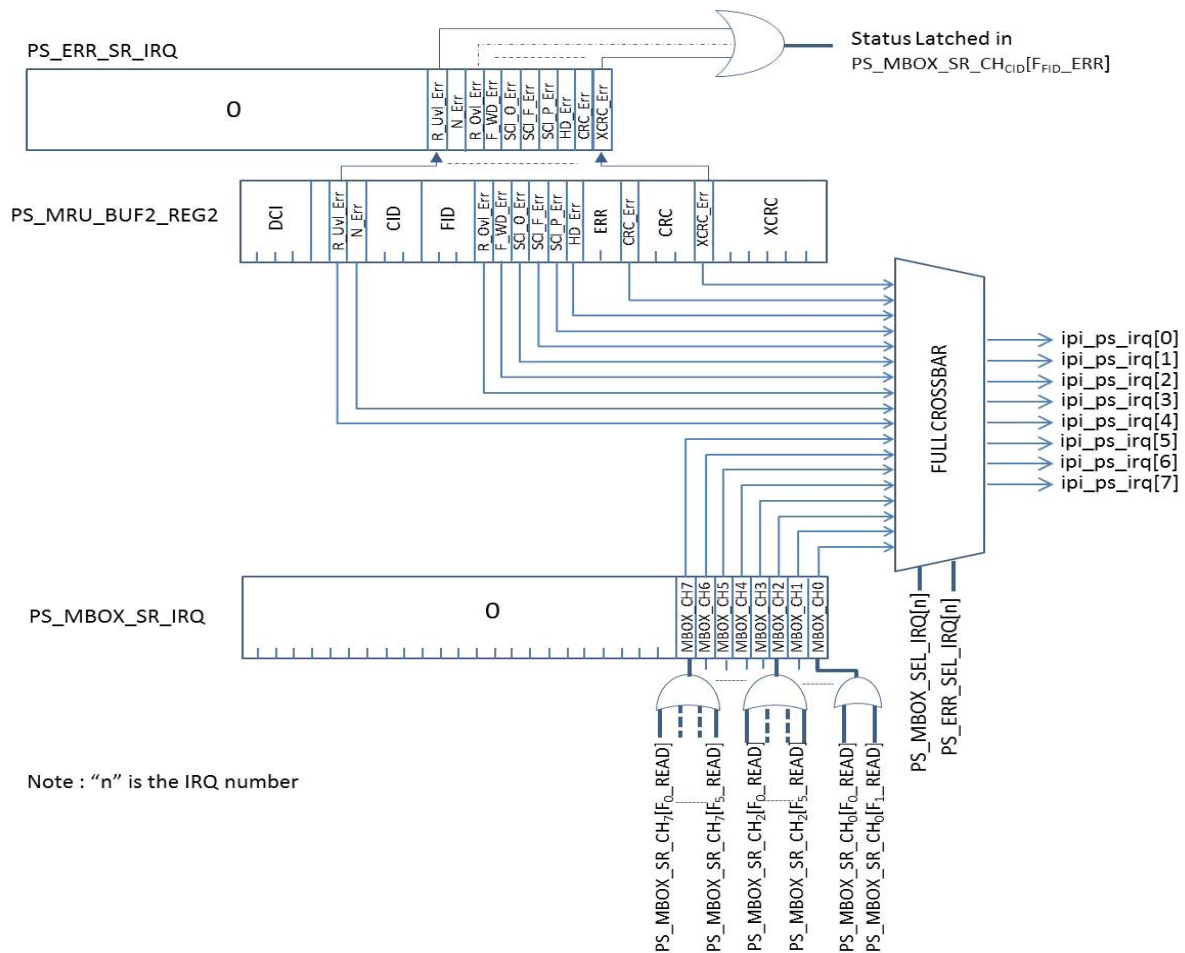
Following are the various interrupt signals that represent the various events:

1. PSI5-S general interrupt request (**Status flags in PS_MBOX_SR_IRQ and PS_ERR_SR_IRQ**)/ipi_ps_irq[irq_n] where irq_n = 0 to 7 and denotes the irq number. There are 8 such interrupts in the IP. Each of these “irq” lines can be configured to respond to any or all of the following events:
 - At least one message of the specific channel has been read by the DMA from the MRU buffer. (The interrupt is configurable to respond to any channel through the corresponding PS_MBOX_SEL_IRQ[irq_n] register)
 - When there is a XCRC error in a new packet that is transferred to MRU buffer2 OR
 - When there is a CRC or parity error in a new packet that is transferred to MRU buffer2 OR
 - When the watchdog Error has been detected in the message just written to MRU buf2 OR
 - When the message header has been detected with transceiver error flags OR
 - When the PSI5 state machine detects in the UART, the message parity error, the overflow error or the framing error and the corresponding frame is transferred to MRU buffer 2.
 - When the “R_OVL_ERR” error occurs in the new packet that is transferred to MRU buffer 2 OR
 - When the “R_UVL_ERR” error occurs in the new packet that is transferred to MRU buffer 2 OR
 - When the number of bytes received does not match the expected Fn bytes (N_ERR == 1)

Each of the “ipi_ps_irq[irq_n]” lines can be configured to be asserted on any of the above events by correctly programming the registers PS_MBOX_SEL_IRQ[irq_n] and PS_ERR_SEL_IRQ[irq_n].

This interrupt is cleared when the corresponding bits in PS_MBOX_SR_IRQ and PS_ERR_SR_IRQ are cleared. It is assumed that one event would not be mapped on more than one “irq” lines. If the same event is mapped on more than one “irq” lines then clearing the corresponding status bit in the PS_MBOX_SR_IRQ or PS_ERR_SR_IRQ will clear all the interrupt requests related to that same event. [Figure 1259](#) shows the interrupt structure for IRQ interrupts.

Figure 1259. IRQ[n] interrupts

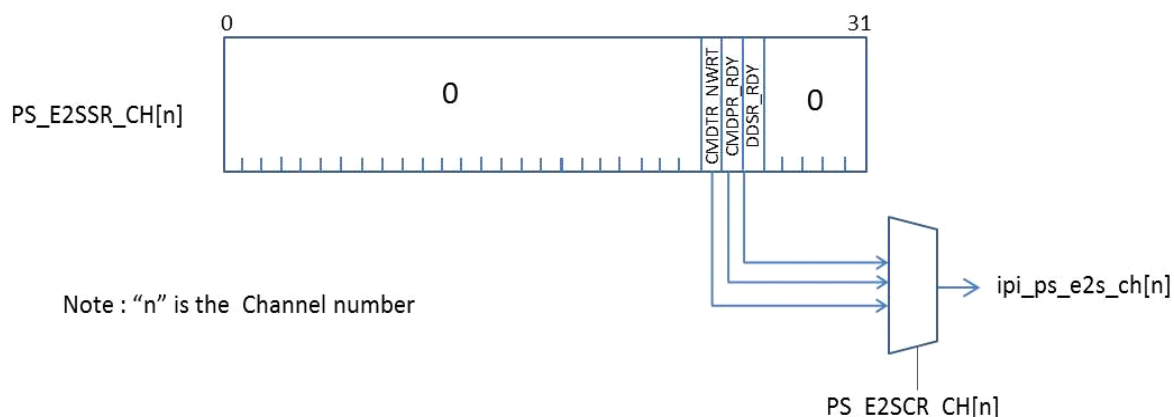


2. PSI5-S ECU-to-Sensor interrupt (**Status Flags in PSI5_E2SSR**)/ipi_ps_e2s_ch[n] where n = 1 to 7:

This interrupt is used for the ECU to Sensor Communications and is generated in the following cases:

- When the DDSR is empty and is ready to accept a new command new from the application. In this case the PS_E2SSR[DDSR_RDY] bit is set to "1".
- When the DDSR is still busy (PS_E2SSR[DDSR_RDY] bit is set to "0") and the application attempts to write a new command to the DDSR. The new command is rejected and the PS_E2SSR[CMD_NWRT] bit is set to "1".
- When the command that is written to the DDSR has finished calculating the CRC/stuff bits/start bits and is ready for transmission the PS_E2SSR[CMDPR_BZY] goes to "1" leading to the generation of this interrupt.

Figure 1260. ECU to Sensor interrupts

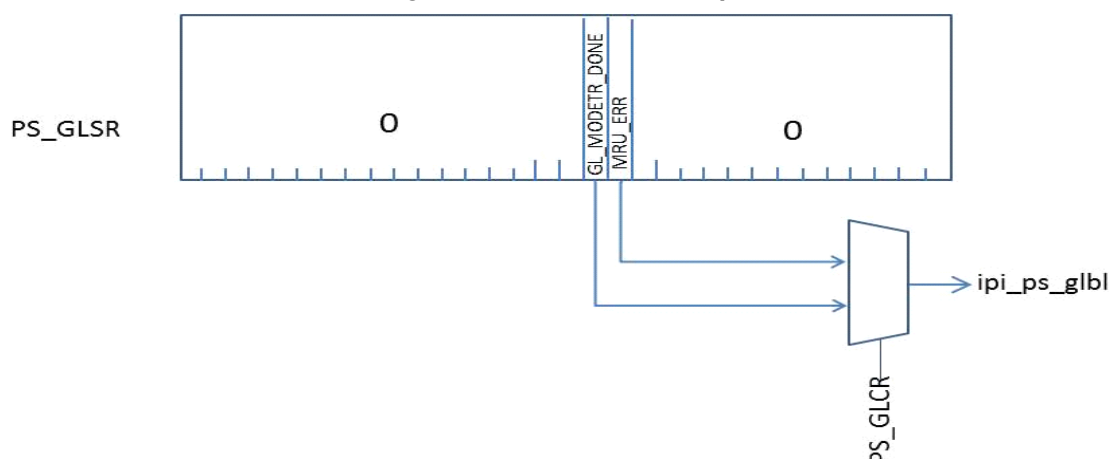


3. Global interrupt (**Status Flags in PS_GLSR**) /ipi_ps_glbl:

This interrupt is a single interrupt for the whole IP and is generated in the following cases.

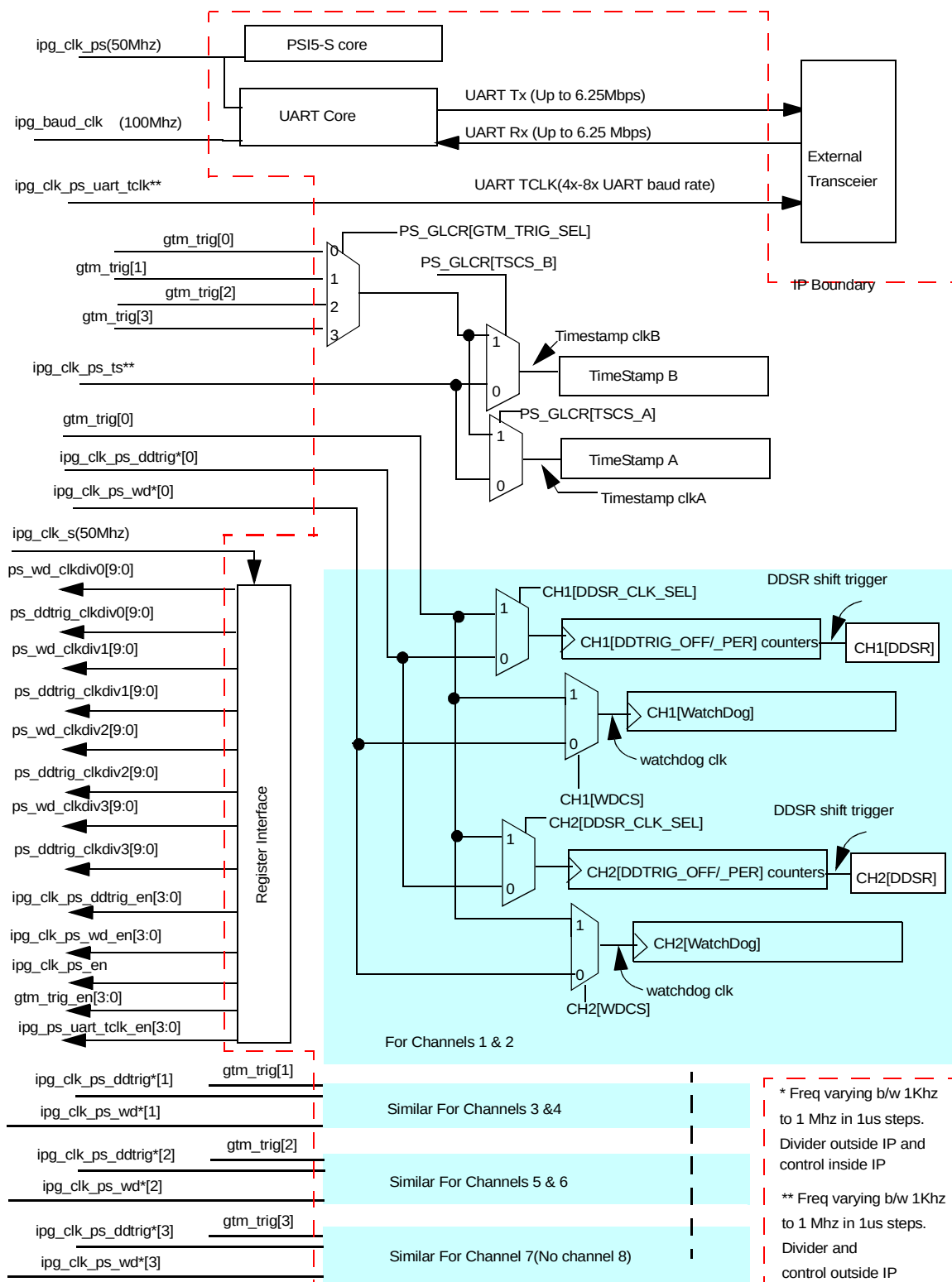
- When the DMA has not completed reading the MRU buffer2; simultaneously there is a pending data in MRU buffer1 that needs to be offloaded to the MRU buffer2 and a new message comes, that overwrites the contents of MRU buffer 1. The channel ID and the Frame ID of the overwritten message can be read from the PS_GLSR registers.
- When there is any change in the “PSI5-S” global modes of the IP . Thus it is asserted during the following transitions UART_STDALONE to PS_DISABLE , PS_DISABLE to PS_CONFIG , PS_DISABLE to PS_NORMAL and PS_CONFIG to PS_NORMAL . Assertion of this interrupt indicates that the desired mode has been accomplished by the IP.

Figure 1261. Global Interrupts



54.5 PSI5-S Clock and Reset

Figure 1262. Clock distribution in PSI5-S



[Figure 1262](#) shows the clocking scheme for the PSI5-S IP. It also shows the distribution of various clocks on the basis of channels .

The divider and the control of clock “ipg_clk_ps_ts” is outside the IP , while for the clocks “ipg_clk_ps_wd” , “ipg_clk_ps_ddtrig” and “ipg_clk_ps_uart_tclk” , though the frequency divider is placed outside the IP yet the value of this divider is controlled from inside IP using the “*_clkdivN” signals which are in turn driven by IPS registers of the IP.

For the safe operation of the UART module , the following relation should be satisfied between the “ipg_baud_clk” and “ipg_clk_ps” :

$$\text{ipg_baud_clk} > \text{ipg_clk_ps} > \text{ipg_baud_clk}/3$$

55 SENT Receiver (SRX)

55.1 Introduction

The Single Edge Nibble Transmission (SENT) Receiver (SRX) module is a multi-channel receiver for receiving serial data frames which are being transmitted by a sensor implementing the SENT encoding scheme and present them to the CPU for further processing.

This module is based on the SAE J2716 information report titled "SENT - Single Edge Nibble Transmission for Automotive Applications" and released on January 27, 2010 (<http://www.sae.org>), Apr2007 and Feb2008. As per this standard, the SENT protocol is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/Ds and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

The SENT encoding scheme is a unidirectional communications scheme from the sensor / transmitting device to the controller /receiving device which does not include a coordination signal from the controller/receiving device. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

55.1.1 Features

The SENT receiver module has the following features:

- The number of SENT channels supported are device-specific. See [Section 7.8.8, SENT receiver \(SRX\) configuration](#) for details.
- Unified message read logic to transfer all messages received to system memory via interrupts or DMA
 - Programmable option per channel to support reading of messages via DMA or Interrupt
 - Internal round robin arbitration sequencer to loop across channels to read messages from each channel leaving CPU/DMA transparent to this process
 - All received messages are stored excluding the synchronization pulse (for Fast messages) and identifiers (for Slow Serial Messages)
 - Separate buffers for storing Fast and Slow Serial Messages
 - Separate DMA and Interrupts generated for Fast and Slow Serial Messages
- Supports compensation for variation in SENT Tx Clock up to $\pm 25\%$ and adjusts its measurement for drift and jitter in the Transmitter and Receiver clocks per channel
 - Calibration pulse correction factor per channel is available for use by software
- Implements for each channel all receiver diagnostics as specified by the SAE Specifications
- Internal 4-bit CRC calculator that can be configured to compute CRC in both Legacy and Recommended method (as given in SAE SENT Specification) for both Fast and Short Serial Message per channel
- Internal 6-bit CRC calculator for Enhanced Serial Messages per channel
- Supports all bit rates by having per channel configurable Rx clock periods from 3 μs to 90 μs

- Support for configurable number of data nibbles per channel
- Option to disable individual channel via programmable control bit
- Time stamping on start of each valid message (i.e. without errors) across all channels. This time stamp is appended to each message for synchronization of messages and for application to understand how much time has elapsed between successive sensor readings
- Support for pause pulse per channel. This can be enabled by programming a bit in the control register for that channel
- Input Programmable Filter on each channel to filter any glitches on input. This filter duration is programmable by the user software.
- 32-bit register interface for programming the module's registers and reading the received messages
- Works on two clocks, one for accurate receive operations (high frequency receiver clock or protocol clock) and other for message read logic (system bus clock) and the module can work irrespective of the frequency relationship between the two clocks. See [Section 55.5, "Clocks and resets"](#) for more details on the clocks.

55.1.2 Modes of operation

55.1.2.1 Debug mode

In debug mode, message reception will be halted and all existing status and messages will be retained. DBG_FRZ bit is required to be set for this mode. On debug exit, a calibration length error (CAL_LEN_ERR) can be reported and the channels will start receiving messages from a valid calibration pulse.

55.1.2.2 Low power modes

SENT Receiver behavior is affected by chip-specific low power modes (for example, STOP or WAIT modes). These low power modes should either gate-off both the system bus clock and the protocol clocks or reduce their frequencies, depending on the application.

On exiting from these low power modes, calibration length error (CAL_LEN_ERR) can get set in channel status registers as the SENT sensor (transmitter) could be in the middle of a message transmission at that point. Thus, software should clear all error status bits after exiting the low power modes.

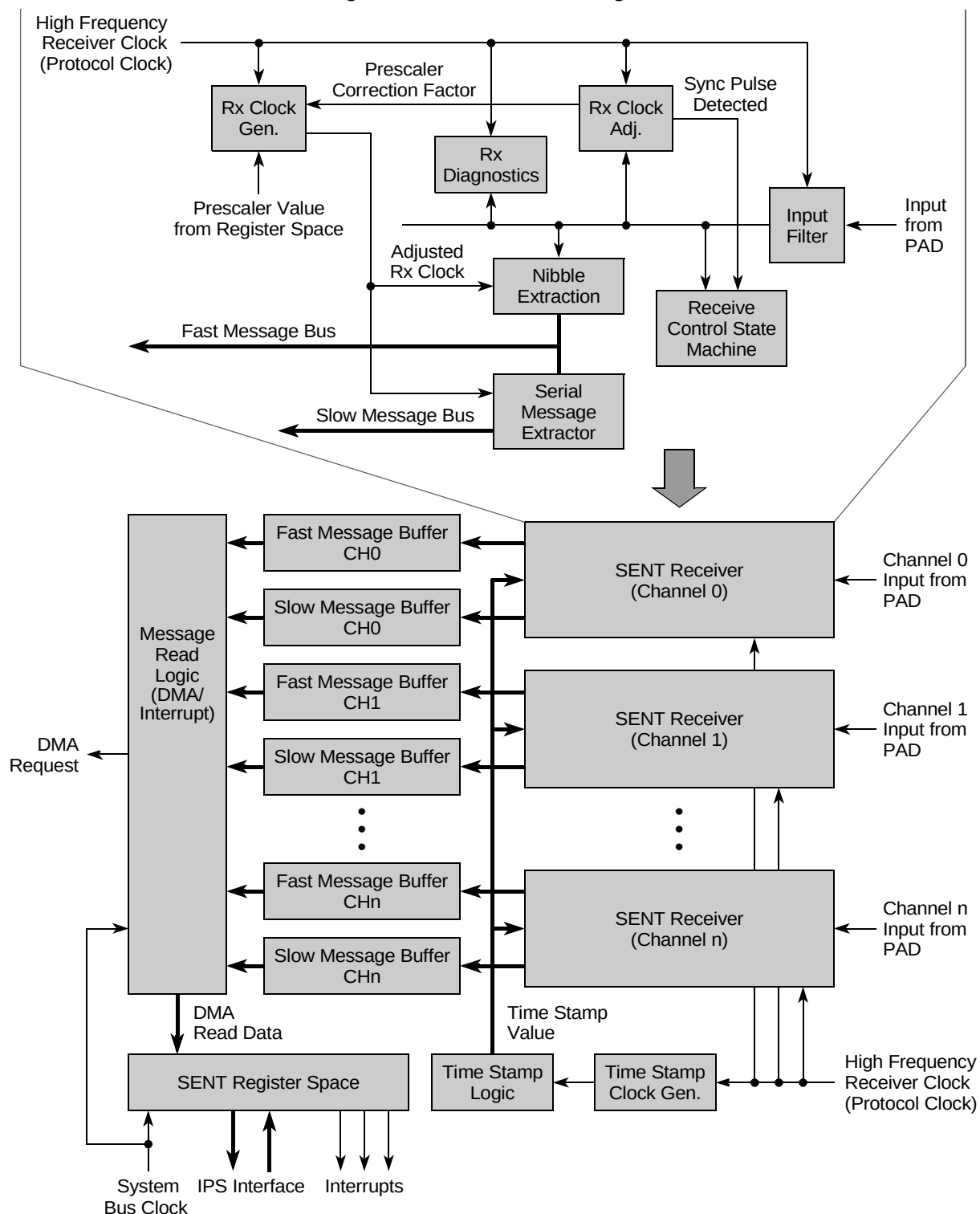
Modes where both system bus clock and protocol clock are gated: In this case, both message reception and interrupt / DMA request generation will halt. The internal receiver channels will freeze (since clock is not there) and will not receive any messages. Messages which are completely received before entering the low power mode can be read after the low power mode exit.

Modes where frequency of system bus clock is changed: For reduced power consumption the system bus clock can be reduced but not below a minimum value (See [Section 55.5.2, "System bus clock requirements"](#)) to avoid overflow of received messages. Protocol clock should not go below the minimum value (See [Section 55.5.3, "High frequency receiver clock \(protocol clock\) requirements"](#)) for accurate receive operations.

Note: Gating of clocks in the low power mode is chip-specific, see the chapter that refers how the modules behave in low power modes.

55.1.3 Block diagram

Figure 1263. SENT block diagram



55.1.4 Design overview

The SENT Receiver Module is a unified receiver module comprising of a number of receiver channels and a unified DMA and Register Space for accessing the messages from each channel. [Figure 1263](#) shows the top level block diagram for the SENT Receiver module.

Each channel of the SENT Receiver is composed of a Clock Generation and Adjustment block to generate the receiver clock and adjust it for frequency variation in the transmitter clock. This block also detects the synchronization/calibration pulse which is present in every SENT message.

The SENT Receiver State Machine controls the sequencing of decoding the messages and combining the information as nibbles. In parallel a diagnostic block is running that monitors the incoming messages for any errors and also verifies the CRC of the received messages. The fast and serial message extractors collate the message nibbles and add the channel number and time stamp information to the messages. A free running time stamp counter is used to add the time stamp value to each message received. The value of this counter is input to each channel and the channel logic appends the time stamp to each message. These messages are then formatted to be stored in the corresponding message read buffer. Only the messages that pass the diagnostics checks are forwarded to be stored in the buffers.

The SENT Receiver provides two mechanisms for reading the received messages, which are via DMA transfers and via Interrupts. Based on the control bit settings in the control registers, a DMA request or Interrupt is asserted to signal the DMA controller or CPU that messages are available to be read.

The free running time stamp counter, register block and DMA interface are common for all channels configured. The register space contains all the necessary registers for all channels. The register space is 32-bit aligned and accessible via 8-/16-/32-bit accesses. The registers for all un-configured channels are marked as reserved.

55.1.4.1 Functional overview

After reset, the SENT module comes up in the disabled state. All channel enable bits and the global enable bit are set to 0 and the user software must program the SENT module parameters correctly before enabling it. The user software must correctly configure all parameters to avoid any underflow or overflow in the message buffers.

Once enabled, the Receiver Clock Generation and Adjustment block detects the synchronization pulse. Until then, the state machine blocks all subsequent operations of the receiver. Once the synchronization pulse is detected, the receiver clock tick period is compensated for variation in transmitter clock tick period and the state machine then waits for the reception of data nibbles. The adjusted receiver clock is used to sample the data nibbles from the incoming serial message. The state machine waits until the status and communication nibble and configured number of data nibbles are received. The state machine then waits for the pause pulse (if configured to receive a pause pulse). The fast and serial message extractor block provides the state machine with the information on how many nibbles have been received. All diagnostics checks are computed in parallel. Once all required data nibbles are received, the CRC is checked, time stamp and channel number is added to the message. The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer along with the message. This final message is stored in the message buffer as described in the

subsequent sections. Any error detected during the receive process causes the status bits in the corresponding channel's status register to be set.

The above process repeats across each enabled channel in parallel and keeps pushing error free messages into the buffers on high frequency receiver clock. The module's message read logic synchronizes all these messages on the bus clock and pops out these messages on any DMA or CPU read access. Any DMA underflow errors are stored in the global status register while the overflow errors are stored in the corresponding channel's status register. In case of an overflow in Fast Messages, the newer messages received will overwrite the previous message.

For detailed description on the complete functionality of each block, see [Section 55.4](#), "[Functional description](#)".

55.2 External signal description

This section lists all inputs to the SENT block.

55.2.1 SENT_RX [(CH-1):0]

These pins serve as the serial data input for each of the SENT channel. There is one bit per channel supported by the device. The SENT sensor input pads on the device using this module should have pull-up enabled to ensure a 1'b1 is seen at the module input when SENT sensor is not driving any data.

Note: The value of the CH is device-specific. See [Section 7.8.8, SENT receiver \(SRX\) configuration](#) for details.

55.3 Memory map and register definition

This section provides a detailed description of all registers accessible in the SENT block.

55.3.1 Memory map

[Table 1227](#) gives an overview on all SENT registers. The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s. The Memory Map and complete module is in Big Endian Format.

The module will not check for correctness of programmed values in Register Space and software must ensure that correct values are being written.

Note: $\text{Register Address} = \text{Base Address} + \text{Address Offset}$, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level. The number of registers are device specific. See the Device Configuration chapter for details.

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

Table 1227. SENT memory map

Offset	Register Name	Access	Reset	Location
0x0000	GBL_CTRL—Global Control Register	RW	0x0000_0010	55.3.2.1/2349
0x0004	CHNL_EN—Channel Enable Register	RW	0x0000_0000	55.3.2.2/2351
0x0008	GBL_STATUS—Global Status Register	W1C	0x0000_0000	55.3.2.3/2352
0x000C	FMSG_RDY—Fast Message Ready Register	RW ⁽¹⁾	0x0000_0000	55.3.2.4/2354
0x0010	SMSG_RDY—Slow Serial Message Ready Register	RW ⁽¹⁾	0x0000_0000	55.3.2.5/2356
0x0014	Reserved			
0x0018	DATA_CTRL1—Data Control Register 1	RW	0x1111_1111	55.3.2.6/2359
0x001C	DATA_CTRL2—Data Control Register 2	RW	0x1111_1111	55.3.2.7/2361
0x0020	Reserved			
0x0024	Reserved			
0x0028	FDMA_CTRL—Fast Message DMA Control Register	RW	0x0000_0000	55.3.2.8/2362
0x002C	SDMA_CTRL—Slow Serial Message DMA Control Register	RW	0x0000_0000	55.3.2.9/2363
0x0030	Reserved			
0x0034	FRDY_IE—Fast Message Ready Interrupt Control Register	RW	0x0000_0000	55.3.2.10/2364
0x0038	SRDY_IE—Slow Serial Message Ready Interrupt Control Register	RW	0x0000_0000	55.3.2.11/2365
0x003C	Reserved			
0x0040	DMA_FMSG_DATA—DMA Fast Message Read Register	R	0x0000_0000	55.3.2.12/2367
0x0044	DMA_FMSG_CRC—DMA Fast Message CRC Read Register	R	0x0000_0000	55.3.2.13/2368
0x0048	DMA_FMSG_TS—DMA Fast Message Time Stamp Read Register	R	0x0000_0000	55.3.2.14/2369
0x004C	Reserved			
0x0050	DMA_SMSG_BIT3—DMA Serial Message Read Register (Bit 3)	R	0x0000_0000	55.3.2.15/2370
0x0054	DMA_SMSG_BIT2—DMA Serial Message Read Register (Bit 2)	R	0x0000_0000	55.3.2.16/2371
0x0058	DMA_SMSG_TS—DMA Serial Message Time Stamp Read Register	R	0x0000_0000	55.3.2.17/2372
0x005C	Reserved			
0x0060 + 0x10 × n	CHn_CLK_CTRL—Channel 'n' Clock Control Register (n = 0 to (CH-1))	RW	0x0000_8000	55.3.2.18/2373
0x0064 + 0x10 × n	CHn_STATUS—Channel 'n' Status Register (n = 0 to (CH-1))	W1C	0x0000_0000	55.3.2.19/2375

Table 1227. SENT memory map (continued)

Offset	Register Name	Access	Reset	Location
0x0068 + 0x10 × n	CHn_CONFIG—Channel 'n' Configuration Register (n = 0 to (CH-1))	RW	0x0000_0104	55.3.2.20/2378
0x006C + 0x10 × n	Reserved			
0x0160 + 0x18 × n	CHn_FMSG_DATA—Channel 'n' Fast Message Register (n = 0 to (CH-1))	R	0x0000_0000	55.3.2.21/2382
0x0164 + 0x18 × n	CHn_FMSG_CRC—Channel 'n' Fast Message CRC Register (n = 0 to (CH-1))	R	0x0000_0000	55.3.2.22/2382
0x0168 + 0x18 × n	CHn_FMSG_TS—Channel 'n' Fast Message Time Stamp Register (n = 0 to (CH-1))	R	0x0000_0000	55.3.2.23/2383
0x016C + 0x18 × n	CHn_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3) (n = 0 to (CH-1))	R	0x0000_0000	55.3.2.24/2383
0x0170 + 0x18 × n	CHn_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2) (n = 0 to (CH-1))	R	0x0000_0000	55.3.2.25/2383
0x0174 + 0x18 × n	CHn_SMSG_TS—Channel 'n' Serial Message Time Stamp Register (n = 0 to (CH-1))	R	0x0000_0000	55.3.2.26/2383
0x02E0	Reserved			

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the [Global Control Register \(GBL_CTRL\)](#)

55.3.2 Register descriptions

This section describes in address order all the SENT registers and their individual bits.

55.3.2.1 Global Control Register (GBL_CTRL)

This register provides the global control and setting for the SENT Receiver Module. The SENT_EN bit must be programmed after all other modules settings and control bits have been written to.

Figure 1264. Global Control Register (GBL_CTRL)

Address Offset = 0x0000

Access: RW

	0	1	2	3	4	5	6	7
R	TSPRSC							
W								
Reset	0	0	0	0	0	0	0	0
	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23
R							FMDUIE	SMDUIE
W								
Reset	0	0	0	0	0	0	0	0
	24	25	26	27	28	29	30	31
R				FAST_CLR		DBG_FRZ		SENT_EN
W								
Reset	0	0	0	1	0	0	0	0

Table 1228. GBL_CTRL field descriptions

Field	Description
0:7 TSPRSC	Time Stamp Prescaler Value. Value in number of clock cycles of high frequency receiver clock set in order to obtain a clock frequency for the time stamp counter block. 0 – Prescaler Bypassed. Time Stamp Clock = High Frequency Receiver Clock 1 to 255 – Value set by user software to obtain a clock period of at least 1μs for the time stamp counter. If the high frequency clock input is 75 MHz, then the value set in this field would be 0x4B (or 75) to obtain a 1μs clock. Note: Since the time stamp clock is common for all channels, the timestamp clock frequency must be at least twice the fastest receiver channel in the application. This is to ensure that the time stamp clock is fast enough to show change in timestamp from message to message.
8:15	Reserved. Read returns zero
16:21	Reserved. Read returns zero

Table 1228. GBL_CTRL field descriptions (continued)

Field	Description
22 FMDUIE	Fast Message DMA Underflow Interrupt Enable: Enables interrupt assertion when there is an underflow condition on Fast Message DMA registers. Default value is 0. 0 – Interrupt is disabled 1 – Interrupt is enabled
23 SMDUIE	Slow Serial Message DMA Underflow Interrupt Enable: Enables interrupt assertion when there is an underflow condition on Slow Message DMA registers. Default value is 0. 0 – Interrupt is disabled 1 – Interrupt is enabled
24	Reserved. Read returns zero
25	Reserved. Read returns zero
26	Reserved. Read returns zero
27 FAST_CLR	Fast Clearing Enable bit. Enables clearing of ready status (for both fast and slow) bit automatically when corresponding message is read. Default value is 1. See Fast Message Ready Status Register (FMSG_RDY) and Slow Serial Message Ready Status Register (SMSG_RDY) for details on the ready status bits. 0 – Fast Clearing is disabled. Bits will be cleared by writing 1 1 – Fast Clearing is enabled
28	Reserved. Read returns zero
29 DBG_FRZ	Debug Freeze. This bit will enable the debug mode support. SENT Module will freeze in debug mode if this bit is set. Default is 0. 0 – No effect in debug mode 1 – Freeze in debug mode
30	Reserved. Read returns zero
31 SENT_EN	SENT Receiver Global Enable. This bit enables or disables the complete receiver module irrespective of the setting of individual channel enable bits. User software must program all channel parameters before setting this bit. Default is 0. 0 – Entire module is disabled 1 – Module is enabled

55.3.2.2 Channel Enable Register (CHNL_EN)

This register is used to enable individual SENT Receiver channels in the module. The user software must ensure all channel parameters and control settings have been programmed before writing to this bit. In case these parameters are changed while the channel is enabled, normal operation is not guaranteed.

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and these bits must be written to 0.

Figure 1265. Channel Enable Register (CHNL_EN)

Address Offset: 0x0004

Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R								
W	EN_CH15	EN_CH14	EN_CH13	EN_CH12	EN_CH11	EN_CH10	EN_CH9	EN_CH8
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R								
W	EN_CH7	EN_CH6	EN_CH5	EN_CH4	EN_CH3	EN_CH2	EN_CH1	EN_CH0
Reset	0	0	0	0	0	0	0	0

Table 1229. CHNL_EN field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 EN_CHn (n = 15 to 0)	Enable bits for Channels 0 to 15. All channels are disabled after reset. User software must program all channel parameters before setting this bit. Once set, other channel parameters in registers should not be changed until this bit is cleared. Only Interrupt Enable and Status bits can be written to. 0 – Channel is disabled 1 – Channel is enabled. Channel parameter registers are locked for writing

55.3.2.3 Global Status Register (GBL_STATUS)

This register provides the status of various conditions in the module that are common across all channels.

Figure 1266. Global Status Register (GBL_STATUS)

Address Offset = 0x0008

Access: W1C

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R							FMDU	SMDU
W							w1c	w1c
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R								
W								
Reset	0	0	0	0	0	0	0	0

Table 1230. GBL_STATUS field descriptions

Field	Description
0:21	Reserved. Read returns zero
22 FMDU	Fast Message DMA Underflow: Interrupt Status bit that indicates when there is an underflow in Fast Message buffers in any channel which has DMA enabled. This bit is set when there is no message in the DMA buffers for read and the software still reads the Fast Message DMA register. Default value is 0. 0 – No underflow 1 – Underflow occurred

Table 1230. GBL_STATUS field descriptions (continued)

Field	Description
23 SMDU	Slow Serial Message DMA Underflow: Interrupt Status bit that indicates when there is an Underflow in Slow Serial Message DMA buffers on channel which has DMA enabled. The bit is set when there is no message in the DMA buffers for read and the software still reads the Slow Serial Message registers. Default value is 0. 0 – No underflow 1 – Underflow occurred
24:31	Reserved. Read returns zero

55.3.2.4 Fast Message Ready Status Register (FMSG_RDY)

The bits of this register are asserted when a Fast Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via [Fast Message DMA Control Register \(FDMA_CTRL\)](#)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via [Fast Message Ready Interrupt Control Register \(FRDY_IE\)](#)) and the channel is enabled, the message ready status bit in this register will get asserted on Fast Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Proper flow cannot be guaranteed if neither is selected.

These bits can be configured to be cleared via “Write 1 to Clear” or “Fast Clearing Mechanism”. The configuration is done by writing to the FAST_CLR bit in the [Global Control Register \(GBL_CTRL\)](#).

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see the Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are inadvisable and may cause unexpected behavior.

Figure 1267. Fast Message Ready Status Register (FMSG_RDY)

Address Offset: 0x000C

Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	F_RDY15	F_RDY14	F_RDY13	F_RDY12	F_RDY11	F_RDY10	F_RDY9	F_RDY8
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	F_RDY7	F_RDY6	F_RDY5	F_RDY4	F_RDY3	F_RDY2	F_RDY1	F_RDY0
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the [Global Control Register \(GBL_CTRL\)](#).

Table 1231. FMSG_RDY field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 F_RDYn (n = 15 to 0)	Fast Message Data Ready Status Bit. Indicates that a Fast Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Fast Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). 0 – No Fast Message is available to be read out via Interrupt 1 – Fast Message is available to be read out via Interrupt The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section Global Control Register (GBL_CTRL) for details on FAST_CLR bit. <i>Note: When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 fast message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported.</i> <i>Note: When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.</i>

55.3.2.5 Slow Serial Message Ready Status Register (SMSG_RDY)

The bits of this register are asserted when a Slow Serial Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via [Fast Message DMA Control Register \(FDMA_CTRL\)](#) and [Slow Serial Message DMA Control Register \(SDMA_CTRL\)](#)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via [Fast Message Ready Interrupt Control Register \(FRDY_IE\)](#) and [Slow Serial Message Ready Interrupt Enable Register \(SRDY_IE\)](#)) and the channel is enabled, the message ready status bit in this register will get asserted on Slow Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Erroneous flow can happen if neither is selected.

These bits can be configured to be cleared via “Write 1 to Clear” or “Fast Clearing Mechanism”. The configuration is done by writing to the FAST_CLR bit in the [Global Status Register \(GBL_STATUS\)](#).

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are inadvisable and may cause unexpected behavior.

Figure 1268. Slow Serial Message Ready Status Register (SMSG_RDY)

Address Offset = 0x0010

Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	S_RDY15	S_RDY14	S_RDY13	S_RDY12	S_RDY11	S_RDY10	S_RDY9	S_RDY8
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	S_RDY7	S_RDY6	S_RDY5	S_RDY4	S_RDY3	S_RDY2	S_RDY1	S_RDY0
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the [Global Control Register \(GBL_CTRL\)](#).

Table 1232. SMSG_RDY field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 S_RDYn (n = 15 to 0)	Slow Serial Message Data Ready Status Bit. Indicates that a Slow Serial Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Serial Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). 0 – No Slow Serial Message is available to be read out via Interrupt 1 – Slow Serial Message is available to be read out via Interrupt The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section Global Control Register (GBL_CTRL) for details on FAST_CLR bit. <i>Note: When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 Fast Message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported.</i> <i>Note: When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.</i>

55.3.2.6 Data Control Register 1 (DATA_CTRL1)

The Data Control Register 1 (DATA_CTRL1) is described below.

Figure 1269. Data Control Register 1 (DATA_CTRL1)

Address Offset = 0x0018

Access: RW

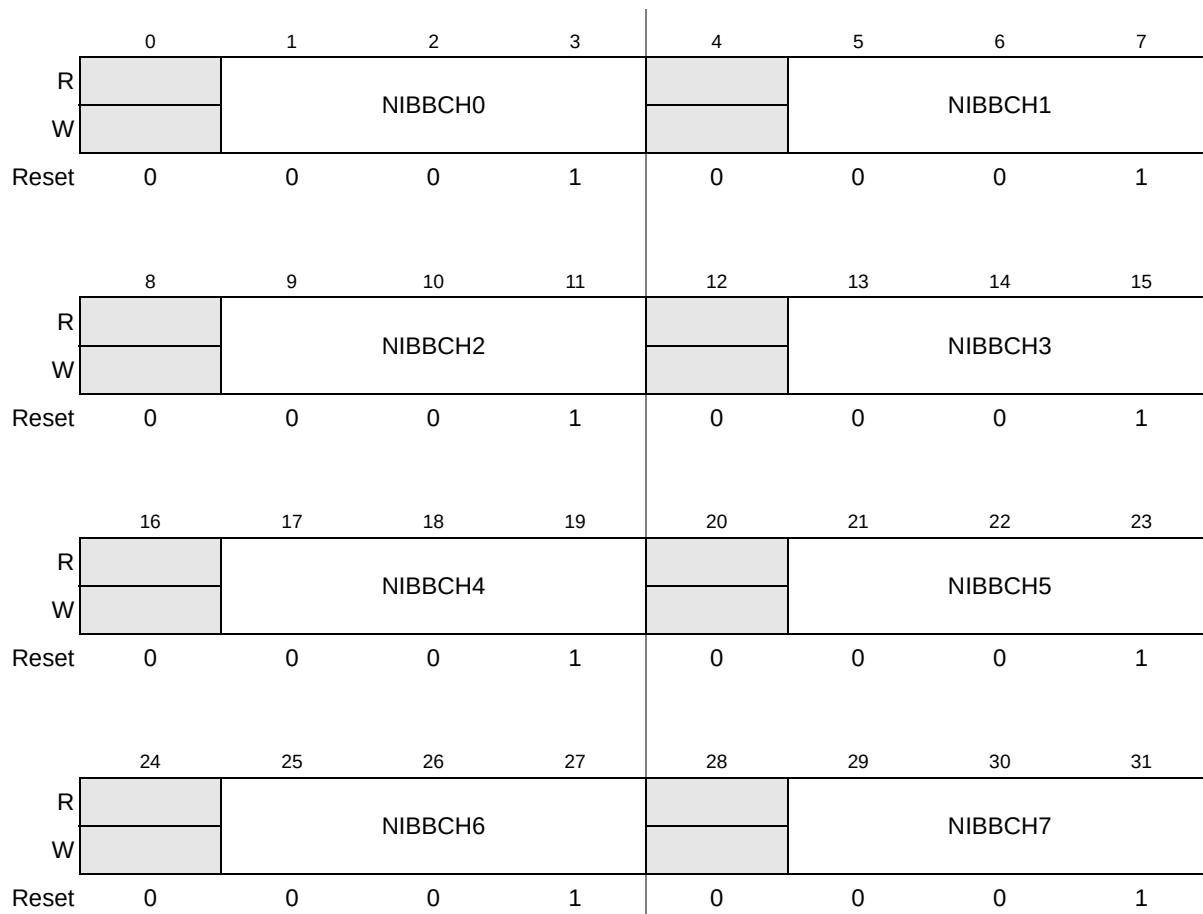


Table 1233. DATA_CTRL1 field descriptions

Field	Description
0	Reserved for future use. Reads returns zeros.
1:3 NIBBCH0	Number of Data Nibbles supported in Channel 0 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
4	Reserved for future use. Reads returns zeros.
5:7 NIBBCH1	Number of Data Nibbles supported in Channel 1 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
8	Reserved for future use. Reads returns zeros.

Table 1233. DATA_CTRL1 field descriptions (continued)

Field	Description
9:11 NIBBCH2	Number of Data Nibbles supported in Channel 2 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
12	Reserved for future use. Reads returns zeros.
13:15 NIBBCH3	Number of Data Nibbles supported in Channel 3 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
16	Reserved for future use. Reads returns zeros.
17:19 NICCH4	Number of Data Nibbles supported in Channel 4 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
20	Reserved for future use. Reads returns zeros.
21:23 NIBBCH5	Number of Data Nibbles supported in Channel 5 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
24	Reserved for future use. Reads returns zeros.
25:27 NIBBCH6	Number of Data Nibbles supported in Channel 6 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
28	Reserved for future use. Reads returns zeros.
29:31 NIBBCH7	Number of Data Nibbles supported in Channel 7 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel

55.3.2.7 Data Control Register 2 (DATA_CTRL2)

Figure 1270. Data Control Register 2 (DATA_CTRL2)

Address Offset = 0x001C

Access: RW

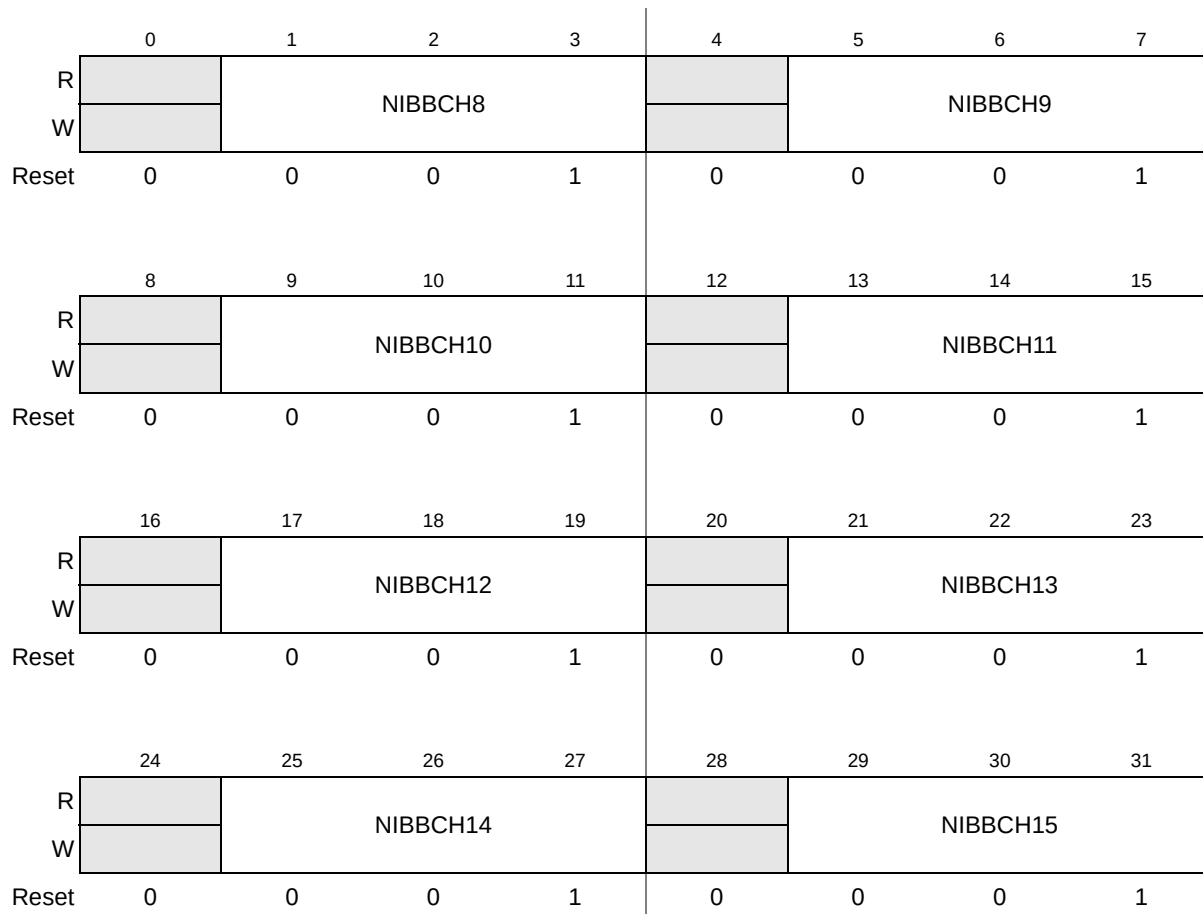


Table 1234. DATA_CTRL2 field descriptions

Field	Description
0	Reserved for future use. Reads returns zeros.
1:3 NIBBCH8	Number of Data Nibbles supported in Channel 8 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
4	Reserved for future use. Reads returns zeros.
5:7 NIBBCH9	Number of Data Nibbles supported in Channel 9 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
8	Reserved for future use. Reads returns zeros.

Table 1234. DATA_CTRL2 field descriptions (continued)

Field	Description
9:11 NIBBCH10	Number of Data Nibbles supported in Channel 10 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
12	Reserved for future use. Reads returns zeros.
13:15 NIBBCH11	Number of Data Nibbles supported in Channel 11 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
16	Reserved for future use. Reads returns zeros.
17:19 NIBBCH12	Number of Data Nibbles supported in Channel 12 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
20	Reserved for future use. Reads returns zeros.
21:23 NIBBCH13	Number of Data Nibbles supported in Channel 13 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
24	Reserved for future use. Reads returns zeros.
25:27 NIBBCH14	Number of Data Nibbles supported in Channel 14 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
28	Reserved for future use. Reads returns zeros.
29:31 NIBBCH15	Number of Data Nibbles supported in Channel 15 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel

55.3.2.8 Fast Message DMA Control Register (FDMA_CTRL)

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are inadvisable and may cause unexpected behavior.

Figure 1271. Fast Message DMA Control Register (FDMA_CTRL)

Address Offset: 0x0028

Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0
	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23
R	FDMA_EN1	FDMA_EN1	FDMA_EN1	FDMA_EN1	FDMA_EN1	FDMA_EN1	FDMA_EN9	FDMA_EN8
W	5	4	3	2	1	0		
Reset	0	0	0	0	0	0	0	0
	24	25	26	27	28	29	30	31
R	FDMA_EN7	FDMA_EN6	FDMA_EN5	FDMA_EN4	FDMA_EN3	FDMA_EN2	FDMA_EN1	FDMA_EN0
W								
Reset	0	0	0	0	0	0	0	0

Table 1235. FDMA_CTRL field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 FDMA_ENn (n = 15 to 0)	Enable DMA for Fast Messages on Channels 0 to 15. These bits are writeable when corresponding Fast Message Ready Interrupt Enable bits are set to 0. 0 – DMA for Fast Messages is disabled 1 – DMA for Fast Messages is enabled

55.3.2.9 Slow Serial Message DMA Control Register (SDMA_CTRL)

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are inadvisable and may cause unexpected behavior.

Figure 1272. Slow Serial Message DMA Control Register (SDMA_CTRL)

Address Offset: 0x002C

Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0
	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23
R	SDMA_EN	SDMA_EN	SDMA_EN	SDMA_EN	SDMA_EN	SDMA_EN	SDMA_EN9	SDMA_EN8
W	15	14	13	12	11	10		
Reset	0	0	0	0	0	0	0	0
	24	25	26	27	28	29	30	31
R	SDMA_EN7	SDMA_EN6	SDMA_EN5	SDMA_EN4	SDMA_EN3	SDMA_EN2	SDMA_EN1	SDMA_EN0
W								
Reset	0	0	0	0	0	0	0	0

Table 1236. SDMA_CTRL field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 SDMA_ENn (n = 15 to 0)	Enable DMA for Slow Serial Messages on Channels 0 to 15. These bits are writeable when corresponding Slow Serial Message Ready Interrupt Enable bits are set to 0. 0 – DMA for Slow Serial Messages is disabled 1 – DMA for Slow Serial Messages is enabled

55.3.2.10 Fast Message Ready Interrupt Control Register (FRDY_IE)

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Figure 1273. Fast Message Ready Interrupt Enable Register (FRDY_IE)

Address Offset: 0x0034

Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	FRDY_IE15	FRDY_IE14	FRDY_IE13	FRDY_IE12	FRDY_IE11	FRDY_IE10	FRDY_IE9	FRDY_IE8
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	FRDY_IE7	FRDY_IE6	FRDY_IE5	FRDY_IE4	FRDY_IE3	FRDY_IE2	FRDY_IE1	FRDY_IE0
W								
Reset	0	0	0	0	0	0	0	0

Table 1237. FRDY_IE field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 FRDY_IEn (n = 15 to 0)	Enable for Fast Message Ready Interrupt on Channels 0 to 15. These bits are writeable when corresponding DMA enable bits are set to 0 in the Fast Message DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Fast Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit is asserted in this register, an interrupt for corresponding channel is generated. 0 – Interrupt on reception of Fast Messages is disabled 1 – Interrupt on reception of Fast Messages is enabled

55.3.2.11 Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels.

Reads of bits beyond the supported number of channels should be ignored and writes to these bits are inadvisable and may cause unexpected behavior.

Figure 1274. Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)

Address Offset = 0x0038

Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0
	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23
R								
W	SRDY_IE15	SRDY_IE14	SRDY_IE13	SRDY_IE12	SRDY_IE11	SRDY_IE10	SRDY_IE9	SRDY_IE8
Reset	0	0	0	0	0	0	0	0
	24	25	26	27	28	29	30	31
R								
W	SRDY_IE7	SRDY_IE6	SRDY_IE5	SRDY_IE4	SRDY_IE3	SRDY_IE2	SRDY_IE1	SRDY_IE0
Reset	0	0	0	0	0	0	0	0

Table 1238. SRDY_IE field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 SRDY_IEn (n = (CH-1) to 0)	Enable for Slow Serial Message Ready Interrupt on Channels 0 to 15. These bits are writeable when corresponding DMA enable bits are set to 0 in the Slow Serial DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Slow Serial Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit in this register is asserted, an interrupt for corresponding channel is generated. 0 – Interrupt on reception of Slow Serial Messages is disabled 1 – Interrupt on reception of Slow Serial Messages is enabled

55.3.2.12 DMA Fast Message Data Read Register (DMA_FMSG_DATA)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_DATA) via DMA.

Figure 1275. DMA Fast Message Data Read Register (DMA_FMSG_DATA)

Address Offset: 0x0040

Access: R

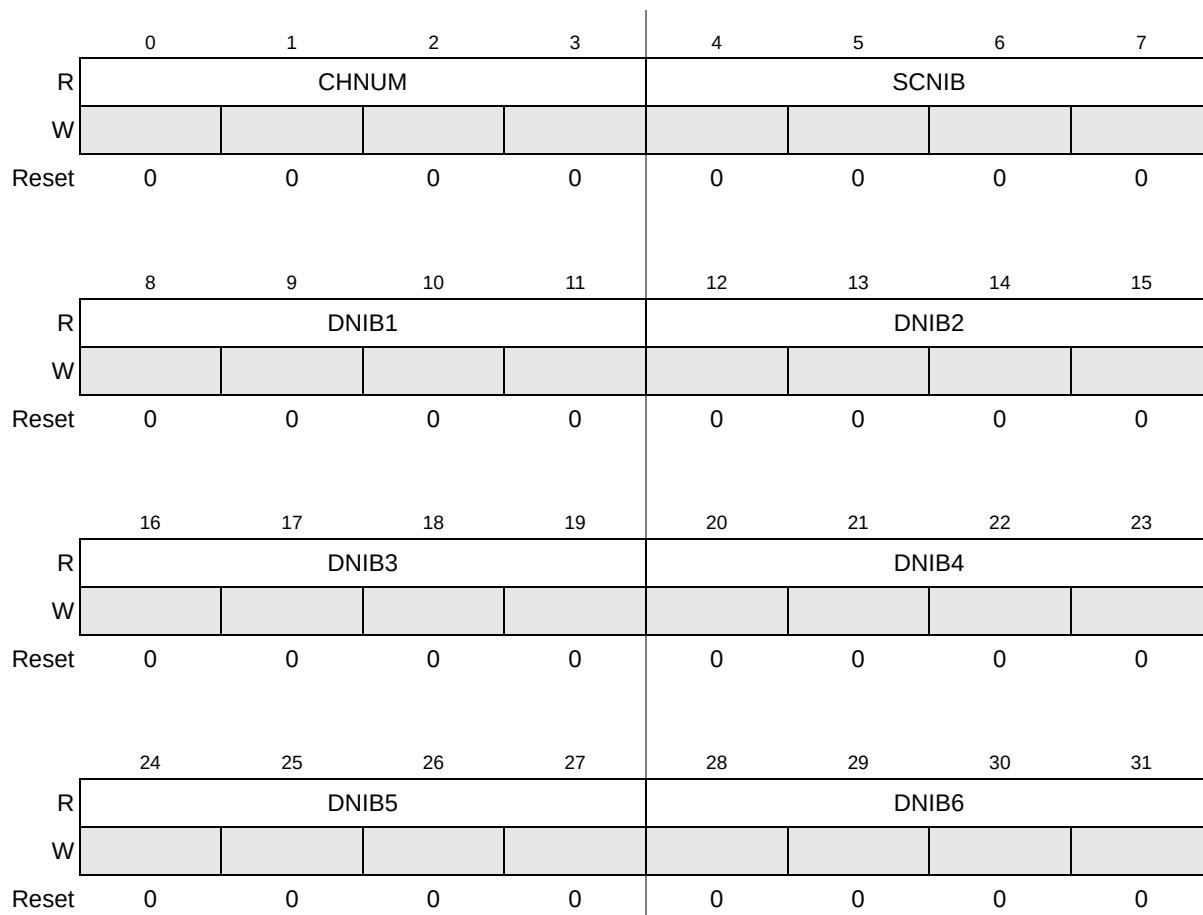


Table 1239. DMA_FMSG_DATA field descriptions

Field	Description
0:3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to '15'
4:7 SCNIB	Status and Communication Nibble of message
8:11 DNIB1	Data Nibble 1. Always supported
12:15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16:19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less

Table 1239. DMA_FMSG_DATA field descriptions (continued)

Field	Description
20:23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24:27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28:31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

55.3.2.13 DMA Fast Message CRC Read Register (DMA_FMSG_CRC)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_CRC) via DMA.

Figure 1276. DMA Fast Message CRC Read Register (DMA_FMSG_CRC)

Address Offset = 0x0044

Access: R

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0
	8	9	10	11	12	13	14	15
R					CRC4b			
W								
Reset	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23
R								
W								
Reset	0	0	0	0	0	0	0	0
	24	25	26	27	28	29	30	31
R								
W								
Reset	0	0	0	0	0	0	0	0

Table 1240. DMA_FMSG_CRC field descriptions

Field	Description
0:11	Reserved. Read returns zeros.
12:15 CRC4b	4-bit CRC value of the message.
16:31	Reserved. Read returns zeros.

55.3.2.14 DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the [Global Control Register \(GBL_CTRL\)](#) register. DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_TS) via DMA.

Figure 1277. DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)

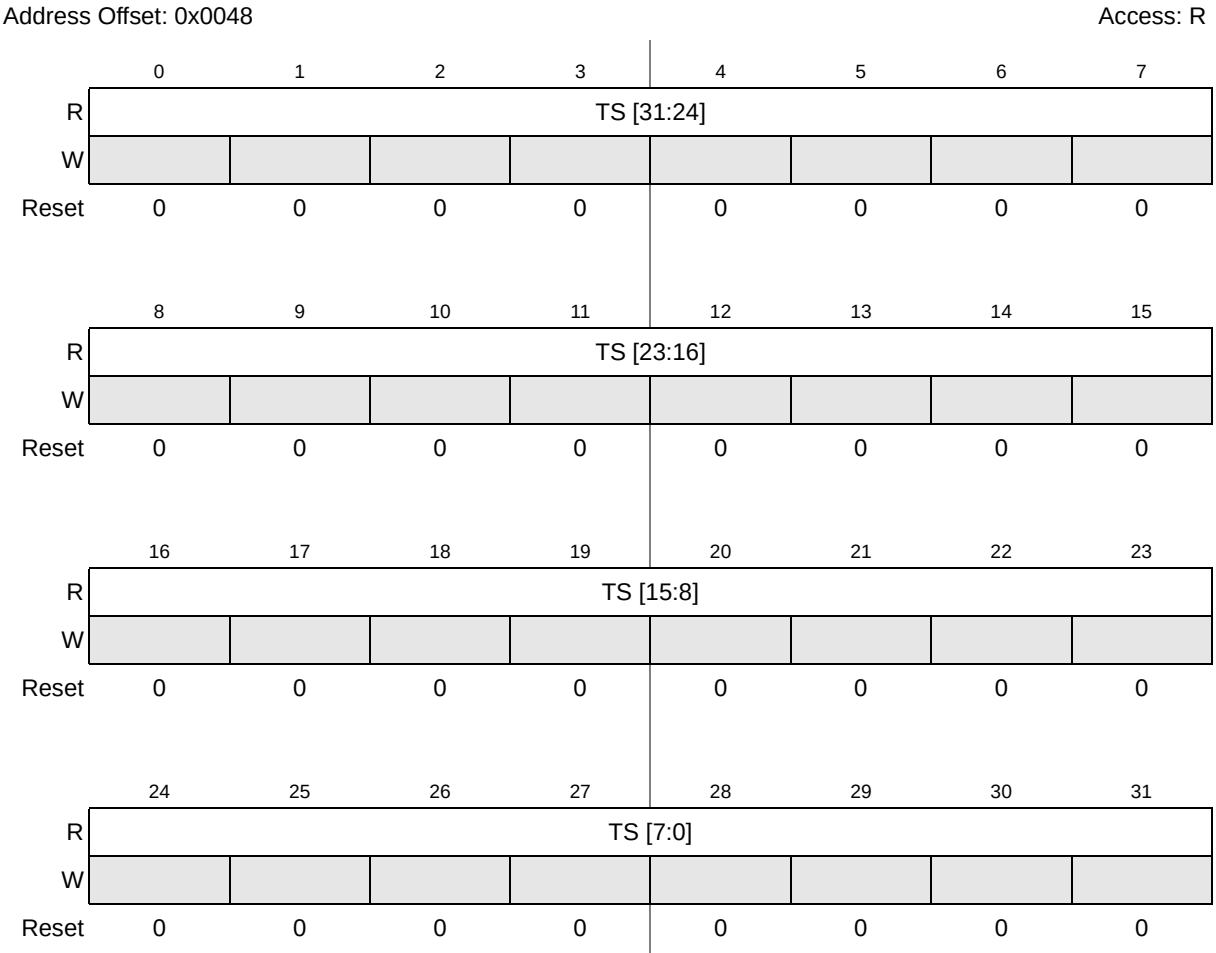


Table 1241. DMA_FMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

55.3.2.15 DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)

The above register (DMA_SMSG_BIT3) is common for all types of slow serial messages. Bit C defines the type of enhanced packet and bit TYPE defines whether it is short serial or enhanced serial message. Since this register is used to read both Short and Enhanced Serial Messages, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown in [Figure 1286](#), [Figure 1287](#) and [Figure 1288](#). [Figure 1286](#) and [Figure 1287](#) shows the enhanced serial messages format to be used when reading this register for these two types of messages. [Figure 1288](#) shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Figure 1278. DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)

Address Offset = 0x0050

Access: R

	0	1	2	3	4	5	6	7
R	CHNUM				TYPE			
W								
Reset	0	0	0	0	0	0	0	0
	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23
R					0	CFG	ID[7:6] / ID[3:2]	
W								
Reset	0	0	0	0	0	0	0	0
	24	25	26	27	28	29	30	31
R	ID[5:4] / ID[1:0]		0	ID[3:0] / DATA[15:12]				0
W								
Reset	0	0	0	0	0	0	0	0

Table 1242. DMA_SMSG_BIT3 Field Descriptions

Field	Description
0:3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to '15'
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 – Short Serial Message 1 – Enhanced Serial Message
5:19	Reserved. Read returns zeros
20	Placed as 0 by Specification
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 – Enhanced Serial Message with 8-bit ID field 1 – Enhanced Serial Message with 4-bit ID field
22:25 ID[7:4]/ID[3:0]	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26	0 as per SAE Specification
7:30 ID[3:0]/DATA[15:12]	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31	0 as per SAE Specification

Note: *If the TYPE bit in the register reads '0', then remainder bits (for CFG, ID & DATA) will be redundant and read zeros. The short serial message will be located in DMA_SMSG_BIT2 register fields. If TYPE bit is '1', then the message in register is Enhanced Serial Message and using CFG bit, software can decode the ID & DATA fields.*

55.3.2.16 DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)

This register (DMA_SMSG_BIT2) is common for all types of slow serial messages. Hence, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown in [Figure 1286](#), [Figure 1287](#) and [Figure 1288](#). [Figure 1286](#) and [Figure 1287](#) shows the enhanced serial messages format to be used when reading this register for these two types of messages. [Figure 1288](#) shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Figure 1279. DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)

Address Offset: 0x0054

Access: R

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R			SMCRC					
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R					DATA[11:8]			
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	DATA[7:0]							
W								
Reset	0	0	0	0	0	0	0	0

Table 1243. DMA_SMSG_BIT2 field descriptions

Field	Description
0:9	Reserved. Read returns zeros.
10:15 SMCRC	6-bit CRC value of the message. When TYPE bit in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16:19	Reserved. Read returns zeros.
20:31 DATA[11:0]	12-bit Data Value

55.3.2.17 DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)

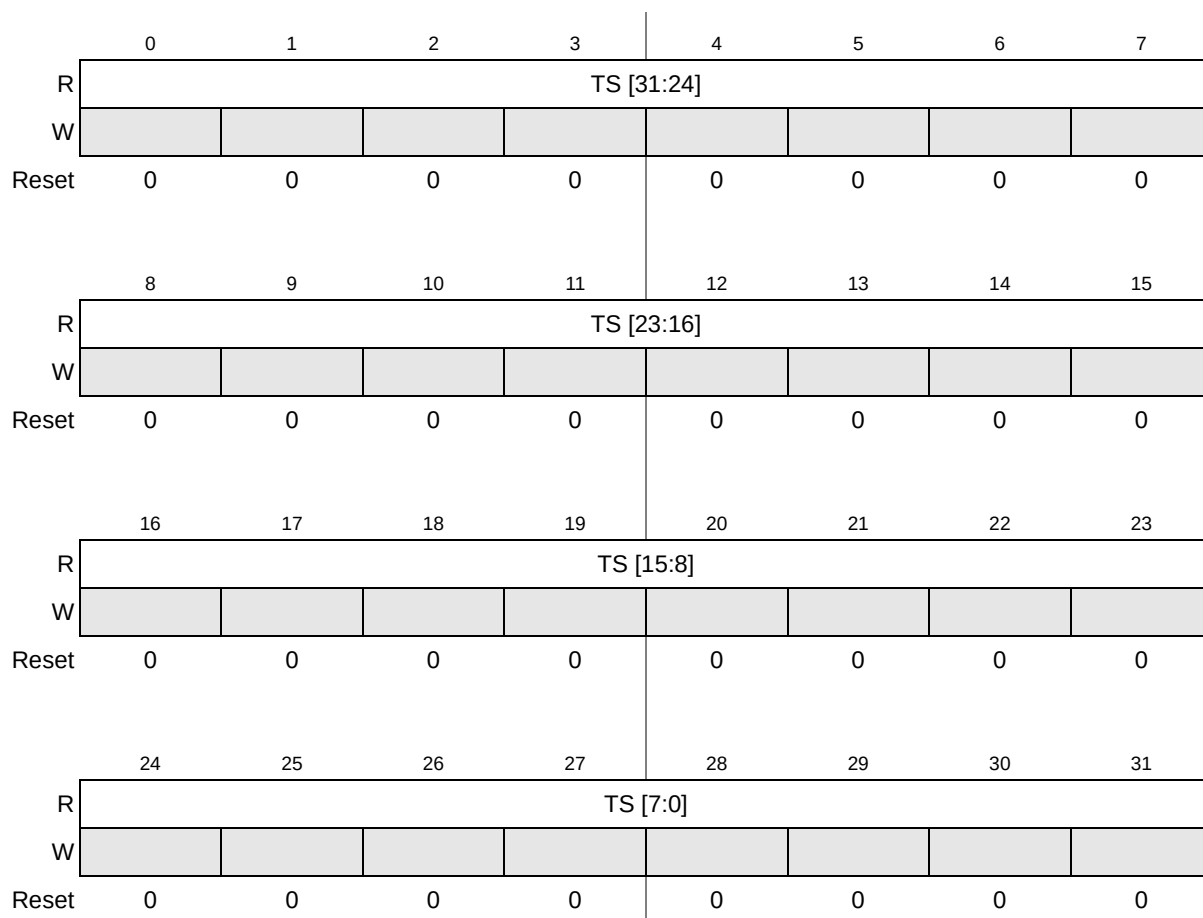
The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the [Global Control Register \(GBL_CTRL\)](#) register.

DMA Source Size must be set to 32 bits when reading this register (DMA_SMSG_TS) via DMA.

Figure 1280. DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)

Address Offset: 0x0058

Access: R

**Table 1244. DMA_SMSG_TS field descriptions**

Field	Description
0:31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

55.3.2.18 Channel 'n' Clock Control Register (n = 0 to (CH-1)) (CHn_CLK_CTRL)

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

The SENT module generates the Receiver Sensor Clock (or simply Receiver Clock) internally to sample the incoming filtered sensor data. The user software must program the prescaler value (for Receiver Clock) to match the ideal Sensor Tx Clock period. The generated Receiver Clock is compensated for variations, clock drift and jitter that might occur in the Sensor Tx Clock. See [Section 55.4.5, "Adjustment for variation in sensor \(Tx\) clock"](#) for more details.

This register configures the generation and subsequent compensation logic of the Receiver Clock. User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the [Channel Enable Register \(CHNL_EN\)](#).

Figure 1281. Channel 'n' Clock Control Register (CHn_CLK_CTRL)

Address Offset = $0x0060 + n \times 0x10$

Access: RW

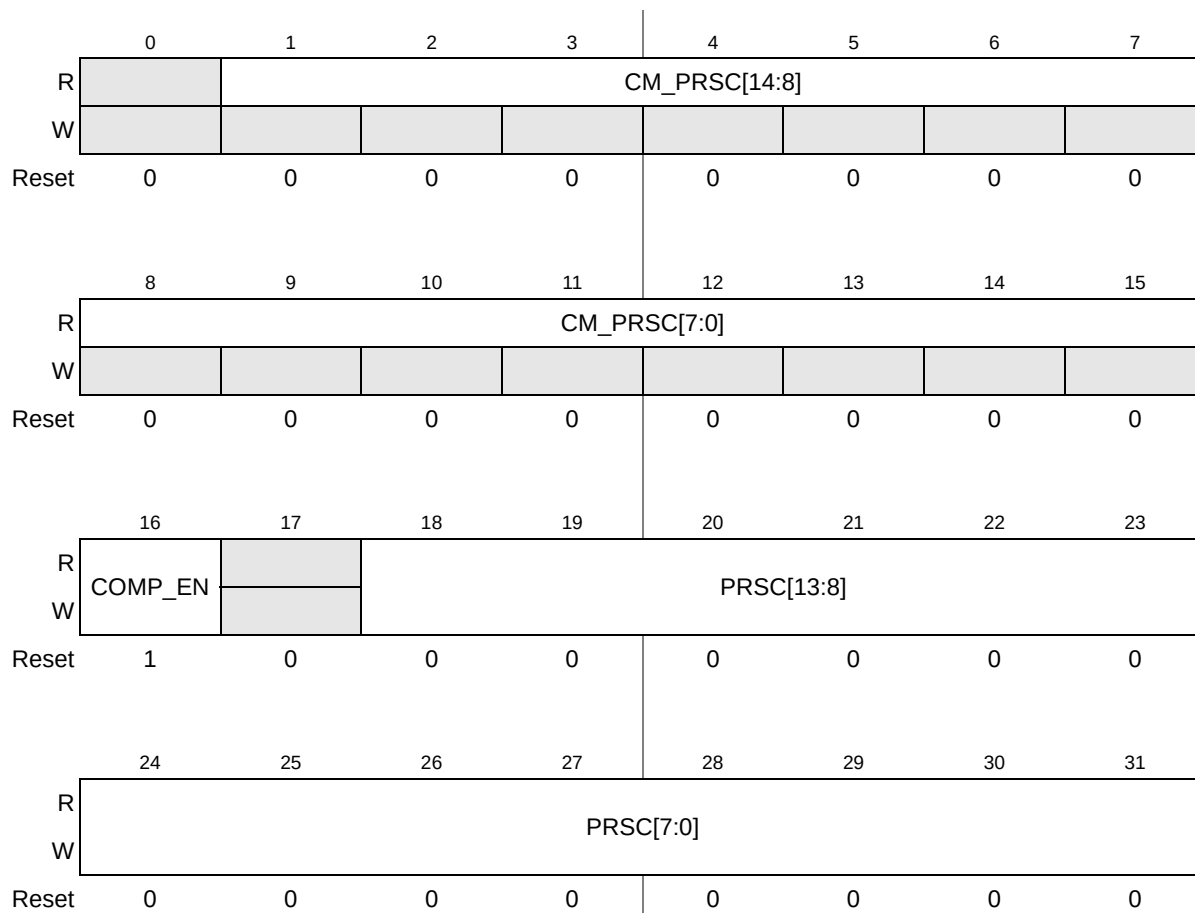


Table 1245. CHn_CLK_CTRL field descriptions

Field	Description
0	Reserved. Read returns zero
1:15 CM_PRSC	Compensated Prescaler Value. This is the compensated prescaler value calculated by Clock Compensation logic and this value is used to compensate for sensor Tx clock variation. This value is indicated as a read only field for use by user software, if needed. This is a 15-bit value. '= Rx Prescaler Value' – No variation, hence no compensation or Compensation is disabled '< Rx Prescaler Value' – Tx Clock is SLOWER than the programmed receive clock '> Rx Prescaler Value' – Tx Clock is FASTER than the programmed receive clock Thus, the new Receiver Clock Tick Period can be computed by software as <i>Compensated Prescaler Value</i> × <i>High Frequency Receiver Clock Period</i>

Table 1245. CHn_CLK_CTRL field descriptions (continued)

Field	Description
16 COMP_EN	Compensation Enable. To enable compensation logic to adjust the receiver clock against variation in Tx clock on this channel. User must set the Prescaler Value before enabling this bit. 0 – Compensation is disabled 1 – Compensation is enabled
17	Reserved. Read returns zero
18:31 PRSC	Rx Prescaler Value. This factor is required to generate required Receiver Clock tick from the High Frequency Receiver Clock. This value can only be written when Channel Enable bit is Zero. Hence this value must be programmed before enabling the channel. If required receiver clock tick period (T_{rx_clk}) is not divisible by the high frequency receiver clock (protocol clock) period (T_{hf_clk}), the required prescaler value will have a fractional part which cannot be programmed. In this case then user is advised to program the prescaler value to the lower integer value. 0 – Prescaler is bypassed. Should not be used. Non-Zero Value – Enables Prescaler The required value is computed by software as follows: Equation 81 $\text{Prescaler_value} = \text{floor}\left(\frac{T_{rx_clk}}{T_{hf_clk}}\right)$ (both clock periods taken in μs). Please see Section 55.5.3, “High frequency receiver clock (protocol clock) requirements before setting this value. For example, if the required receiver clock tick period is 43 μs and the High Frequency Receiver clock is selected to be 62.5 MHz (or 0.016 μs) then, Prescaler Value = 43/0.016 = 2687.5. Hence user must program 2687 in this register.

55.3.2.19 Channel ‘n’ Status Register (n = 0 to (CH-1)) (CHn_STATUS)

This registers stores information about certain events and error conditions that occur in the channel during the message reception process. The assertion of these bit is not linked to any message but it is a general alert to the CPU that a particular condition (error or normal event) happened in the recent past. Messages are automatically discarded when an error is detected.

User software can clear these bit at any time by writing 1 to them. This register can be updated at any time irrespective of the setting in the [Channel Enable Register \(CHNL_EN\)](#) bit for the corresponding channel.

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

Figure 1282. Channel ‘n’ Status Register (CHn_STATUS)

Address Offset = 0x0064 + n × 0x10

Access: W1C

	0	1	2	3	4	5	6	7
R	BUS_IDLE				CAL_RESYNC	CAL_20_25	SMSG_OFLW	FMSG_OFLW
W	w1c				w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R		PP_DIAG_ERR	CAL_LEN_ERR	CAL_DIAG_ERR	NIB_VAL_ERR	SMSG_CRC_ERR	FMSG_CRC_ERR	NUM_EDGES_ERR
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R								
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R								
W								
Reset	0	0	0	0	0	0	0	0

Table 1246. CHn_STATUS field descriptions

Field	Description
0 BUS_IDLE	Bus Idle Status. This bit indicates that the sensor interface has been idle for more than the period defined by the programmed Bus Idle Count value (in Channel 'n' Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)). CPU should write 1 to this clear this bit once it is read. 0 – Bus is not idle 1 – Channel has been idle for more than the allowed value
1:3	Reserved. Read returns zero
4 CAL_RESYNC	Successive Calibration Check (Option 2 of SAE SENT Spec) Resynchronized. This bit indicates that Successive Calibration Check (Option 2 of SAE Spec) has failed three times and has resynchronized to assume 3rd message to be correct and will be received. The successive calibration check error will also be asserted during re-synchronization (to indicate the 3rd error being detected). 0 – No interrupt 1 – Interrupt Status condition has occurred
5 CAL_20_25	Calibration Variation 20 – 25% Interrupt Status. This bit indicates that the calibration pulse received on this channel has variation in between $\pm 20\%$ to $\pm 25\%$ from 56 ticks. 0 – No Interrupt 1 – Interrupt Status condition has occurred
6 MSG_OFLW	Slow Serial Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow will cause the older messages (which are not read out) to be overwritten with newly received messages. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 – No Interrupt 1 – Interrupt Status condition has occurred
7 FMSG_OFLW	Fast Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow happens when effective data read rate by CPU is slower than data receive rate on channel. On overflow, new incoming messages will overwrite the previous messages (that are not read out) stored in the receiver buffers. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 – No Interrupt 1 – Interrupt Status condition has occurred
8	Reserved. Read returns zero

Table 1246. CHn_STATUS field descriptions (continued)

Field	Description
9:15	Diagnostic Checks Status bits. These bits indicate which diagnostic has failed or passed on this channel. These bits are cleared by writing 1 to them. NUM_EDGES_ERR: Not the expected number of negative edges between calibration pulse. This bit will not set in case of “Option 2” for successive calibration pulse check is selected. This error will only set when “Option 1” for successive calibration check is selected (SUCC_CAL_CHK). FMSG_CRC_ERR: Checksum error in Fast Message SMSG_CRC_ERR: Checksum error in Slow Serial Message NIB_VAL_ERR: Any nibble data value <0 or >15 CAL_DIAG_ERR: Successive Calibration pulses differ by $\pm 1.56\%$. SAE spec defines it be 1.5625% but the accuracy of this check depends on the frequency of protocol clock (See Section 55.5.3, “High frequency receiver clock (protocol clock) requirements). CAL_LEN_ERR: This bit indicates that Calibration pulse is more than 56 ticks $\pm 25\%$. PP_DIAG_ERR: This bit indicates that the ratio of calibration pulse length to overall message length (with pause pulse) is more than $\pm 1.5625\%$ between two messages. This check is valid only for messages with pause pulse. 0 – Error Check has passed 1 – Error Check has failed
15:016:31	Reserved. Read returns zero

55.3.2.20 Channel ‘n’ Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained in [Channel ‘n’ Status Register \(n = 0 to \(CH-1\)\) \(CHn_STATUS\)](#) and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the [Channel Enable Register \(CHNL_EN\)](#).

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

Figure 1283. Channel 'n' Configuration Register (CHn_CONFIG)

Address Offset = $0x0068 + n \times 0x10$

Access: RW

	0	1	2	3	4	5	6	7
R	BUS_IDLE_CNT				IE_CAL_RESYNC	IE_CAL_20_25	IE_SMSG_OFLW	IE_FMSG_OFLW
W								
Reset	0	0	0	0	0	0	0	0
	8	9	10	11	12	13	14	15
R	FCRC_CHK_OFF	IE_PP_DIAG_ERR	IE_CAL_LEN_ERR	IE_CAL_DIAG_ERR	IE_NIB_VAL_ERR	IE_SMSG_CRC_ERR	IE_FMSG_CRC_ERR	IE_NUM_EDGES_ERR
W								
Reset	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23
R	DCHNG_INT	CAL_RNG	PP_CHKSEL	FCRC_TYPE	FCRC_SC_EN	SCRC_TYPE	PAUSE_EN	SUCC_CAL_CHK
W								
Reset	0	0	0	0	0	0	0	1
	24	25	26	27	28	29	30	31
R	FIL_CNT[7:0]							
W								
Reset	0	0	0	0	0	1	0	0

Table 1247. CHn_CONFIG field descriptions

Field	Description
0:3 BUS_IDEL_CNT	Bus Idle Count: This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000: Disabled. 0001: 127*2 Receiver Clock Tick Counts 0010: 127*4 Receiver Clock Tick Counts 0100: 127*8 Receiver Clock Tick Counts 1000: 127*16 Receiver Clock Tick Counts
4 IE_CAL_RESYNC	Successive Calibration Check Resynchronized Interrupt Enable: This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of "Option 2" being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 – Interrupt is disabled 1 – Interrupt is enabled
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable. This bit enables interrupt assertion when corresponding status bit is set. 0 – Interrupt is disabled 1 – Interrupt is enabled
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Slow Serial Messages in corresponding channel 0 – Interrupt is disabled 1 – Interrupt is enabled
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Fast Messages in corresponding channel. 0 – Interrupt is disabled 1 – Interrupt is enabled
8 FCRC_CHK_OFF	Fast Message CRC Check Off: This bit can be used to switch off CRC check in Fast Message 0 – Check is enabled 1 – Check is disabled/off

Table 1247. CHn_CONFIG field descriptions (continued)

Field	Description
9:15	Diagnostic Checks Interrupt Enable bits. These bits enables interrupts assertion when respective diagnostic checks fail on corresponding channel. IE_NUM_EDGES_ERR: Not the expected number of negative edges between calibration pulse IE_FMSG_CRC_ERR: Checksum error in Fast Message IE_SMSG_CRC_ERR: Checksum error in Slow Serial Message IE_NIB_VAL_ERR: Any nibble data value <0 or >15 IE_CAL_DIAG_ERR: Successive Calibration pulses differ by more than $\pm 1.56\%$. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55% IE_CAL_LEN_ERR: Calibration pulse is wider than 56 ticks $\pm 25\%$ IE_PP_DIAG_ERR: Ratio of calibration pulse length to message length varies by more than $\pm 1.5625\%$ between two frames. Valid for messages with pause pulse. Configuration option for each bit as below: 0 – Interrupt is disabled 1 – Interrupt is enabled
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data: This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 – All Fast Messages will be received even if data nibbles are same 1 – Only Fast Messages with differing values of data nibbles will be received
17 CAL_RNG	Valid Calibration Pulse Range Selection: This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 – 20% variation is acceptable 1 – 25% variation is acceptable
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection: This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 – Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run 1 – Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.
19 FCRC_TYPE	Fast Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 – XOR based implementation (Recommended by SAE Specification) 1 – Legacy LUT based implementation

Table 1247. CHn_CONFIG field descriptions (continued)

Field	Description
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable. This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast Messages. 0 – Status and Communication Nibble <u>not</u> included in CRC calculation 1 – Status and Communication Nibble is included in CRC calculation
21 SCRC_TYPE	Slow Serial Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 – XOR based implementation (Recommended by SAE Specification) 1 – Legacy LUT based implementation
22 PAUSE_EN	Pause Pulse Enable. Enables the channel receiver to detect a pause pulse. 0 – Detection of Pause Pulse is disabled 1 – Detection of Pause Pulse enabled
23 SUCC_CAL_CHK	Successive Calibration Pulse Check Method. This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 – Option 2 i.e. Low Latency Option as per SAE Specification 1 – Option 1 i.e. Preferred but High Latency Option as per SAE Specification
24:31 FIL_CNT	Input Filter Sample Count. This indicates the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'. 0 – No filtering. Input from device's pad is only synchronized Non-Zero – Filtering is enabled

55.3.2.21 Channel 'n' Fast Message Data Read Register (n = 0 to (CH-1)) (CHn_FMSG_DATA)

Address offset: $0x0160 + 0x18 \times n$

This is register CHn_FMSG_DATA. The contents of this register are same as [DMA Fast Message Data Read Register \(DMA_FMSG_DATA\)](#).

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

55.3.2.22 Channel 'n' Fast Message CRC Read Register (n = 0 to (CH-1)) (CHn_FMSG_CRC)

Address offset: $0x0164 + 0x18 \times n$

This is register CHn_FMSG_CRC. The contents of this register are same as [DMA Fast Message CRC Read Register \(DMA_FMSG_CRC\)](#).

55.3.2.23 Channel 'n' Fast Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_FMSG_TS)

Address offset: $0x0168 + 0x18 \times n$

This is register CHn_FMSG_TS. The contents of this register are same as [DMA Fast Message Time Stamp Read Register \(DMA_FMSG_TS\)](#).

55.3.2.24 Channel 'n' Serial Message Read Register (Bit 3) (n = 0 to (CH-1)) (CHn_SMSG_BIT3)

Address offset: $0x016C + 0x18 \times n$

This is register CHn_SMSG_BIT3. The contents of this register are same as [DMA Slow Serial Message Bit3 Read Register \(DMA_SMSG_BIT3\)](#).

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

55.3.2.25 Channel 'n' Serial Message Read Register (Bit 2) (n = 0 to (CH-1)) (CHn_SMSG_BIT2)

Address offset: $0x0170 + 0x18 \times n$

This is register CHn_SMSG_BIT2. The contents of this register are same as [DMA Slow Serial Message Bit2 Read Register \(DMA_SMSG_BIT2\)](#).

55.3.2.26 Channel 'n' Serial Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_SMSG_TS)

Address offset: $0x0174 + 0x18 \times n$

This is register CHn_SMSG_TS. The contents of this register are same as [DMA Slow Serial Message Time Stamp Read Register \(DMA_SMSG_TS\)](#).

55.3.3 Register Classification for Safety

The registers of this module are classified as per Safety Requirements as follows:

Safety Critical: There are no safety critical registers in this module.

Safety Relevant: The register bits/fields in this category are safety relevant as any random error in these bits will lead to unwanted interference to CPU/DMA. The registers containing these bits must be protected by periodic CRC or equivalent check implemented at SoC Level. The following register bits/fields are classified in this category:

- [Global Control Register \(GBL_CTRL\)](#)
- [Channel Enable Register \(CHNL_EN\)](#)
- [Fast Message Ready Status Register \(FMSG_RDY\)](#)
- [Slow Serial Message Ready Status Register \(SMSG_RDY\)](#)
- [Data Control Register 1 \(DATA_CTRL1\)](#) and [Data Control Register 2 \(DATA_CTRL2\)](#)
- [Fast Message DMA Control Register \(FDMA_CTRL\)](#)
- [Slow Serial Message DMA Control Register \(SDMA_CTRL\)](#)
- [Fast Message Ready Interrupt Control Register \(FRDY_IE\)](#)
- [Slow Serial Message Ready Interrupt Enable Register \(SRDY_IE\)](#)
- Fast Message DMA Read Register Set and Slow Message DMA Read Register Set

Safety Latent: The register bits/fields in this category should be protected by LBIST at SoC level. The following register bits/fields are classified in this category:

- [Global Status Register \(GBL_STATUS\)](#)
- [Channel 'n' Status Register \(n = 0 to \(CH-1\)\) \(CHn_STATUS\)](#)
- [Channel 'n' Configuration Register \(n = 0 to \(CH-1\)\) \(CHn_CONFIG\)](#)

Non-Safety: The following register bits are classified in this category:

- All other registers and bits fall in this category

55.4 Functional description

55.4.1 Initialization sequence

All the channels are required to be configured before enabling them by writing into corresponding channel enable bit in the [Channel Enable Register \(CHNL_EN\)](#). There are some registers which must be configured and some which are to be changed if reset value is not as required for that channel. After enabling the channel, no configuration should be changed and if required to be changed, the user needs to first disable the channel and then reconfigure the channel parameters. The following must to be configured before enabling the corresponding channel:

- Time Stamp Prescaler Value in [Global Control Register \(GBL_CTRL\)](#) to generate the time stamping clock
- Rx Prescaler Value in [Channel 'n' Clock Control Register \(n = 0 to \(CH-1\)\) \(CHn_CLK_CTRL\)](#) to generate the channel's receiver clock
- Compensation enable in [Channel 'n' Clock Control Register \(n = 0 to \(CH-1\)\) \(CHn_CLK_CTRL\)](#)
- Channel Configuration settings in the [Channel 'n' Configuration Register \(n = 0 to \(CH-1\)\) \(CHn_CONFIG\)](#)
- Number of Data Nibbles in Channel in the [Data Control Register 1 \(DATA_CTRL1\)](#) and [Data Control Register 2 \(DATA_CTRL2\)](#)
- Control bits that control whether messages received in channel are to read by interrupt or DMA in [Fast Message DMA Control Register \(FDMA_CTRL\)](#), [Slow Serial Message DMA Control Register \(SDMA_CTRL\)](#), [Fast Message Ready Interrupt Control Register \(FRDY_IE\)](#) and [Slow Serial Message DMA Control Register \(SDMA_CTRL\)](#)

55.4.2 DMA read logic

The SENT module generates separate DMA requests for all Fast Messages and Slow Serial Messages received. The DMA requests are common for all channels. Each packet stored in the channels' buffers are appended with a time stamp taken from a free running time keeping counter and the channel number on which it was received. Details on this time stamp counter can be found in [Section 55.4.8, "Time stamp logic"](#).

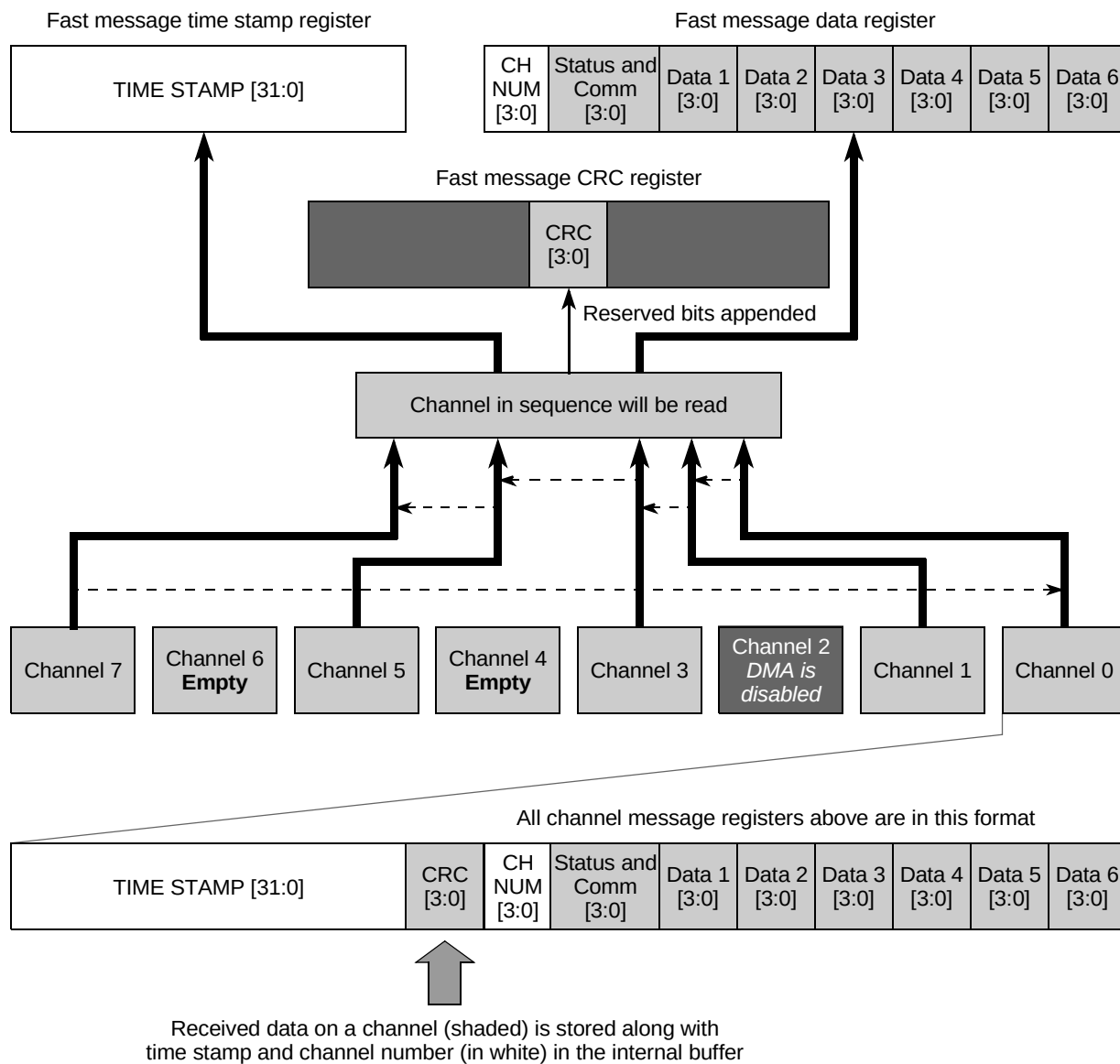
55.4.2.1 DMA request for Fast Message reading

Fast Messages received are stored in a single system bus clock domain buffer to be read out by the DMA Controller via the DMA Fast Message Read register set. Note the DMA

Controller must read all 3 DMA Read Registers to obtain the complete message. The data flow will happen as follows:

- A DMA request will be asserted when any of the DMA enabled channels has successfully received one complete Fast Message. The request is kept asserted till all messages (across DMA enabled channels) have been read out.
- DMA Controller should read the complete message by accessing each register of the DMA Fast Message Read register set, the registers being:
 - [DMA Fast Message Data Read Register \(DMA_FMSG_DATA\)](#)
 - [DMA Fast Message CRC Read Register \(DMA_FMSG_CRC\)](#)
 - [DMA Fast Message Time Stamp Read Register \(DMA_FMSG_TS\)](#)
- A fixed round robin sequencing runs, as shown in [Figure 1284](#), and this logic checks for messages in sequence 0→1→3→5→7 and back to channel 0 and so on. Channel 2 is skipped as it's not DMA enabled) and channel 4 and 6 are skipped as message is not received completely. This sequence continues like this.
- The sequence in which messages are read through DMA interface is not deterministic and depends on the sequence in which they are completely received.
- When the sequencing logic detects a message ready to be read, it will update the DMA Fast Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 read accesses to the above mentioned registers)
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- User software should ensure there is no overflow in channels. In case overflow occurs, the message in the channel buffer will get overwritten by the newly received message. However, when a DMA read is in progress, the buffers will not be overwritten if overflow occurs to maintain data integrity.
- The Round Robin sequencing logic loops through each DMA enabled channel and halts at any channel having data till it is read out by the DMA Controller.

Figure 1284. Fast Message DMA read logic



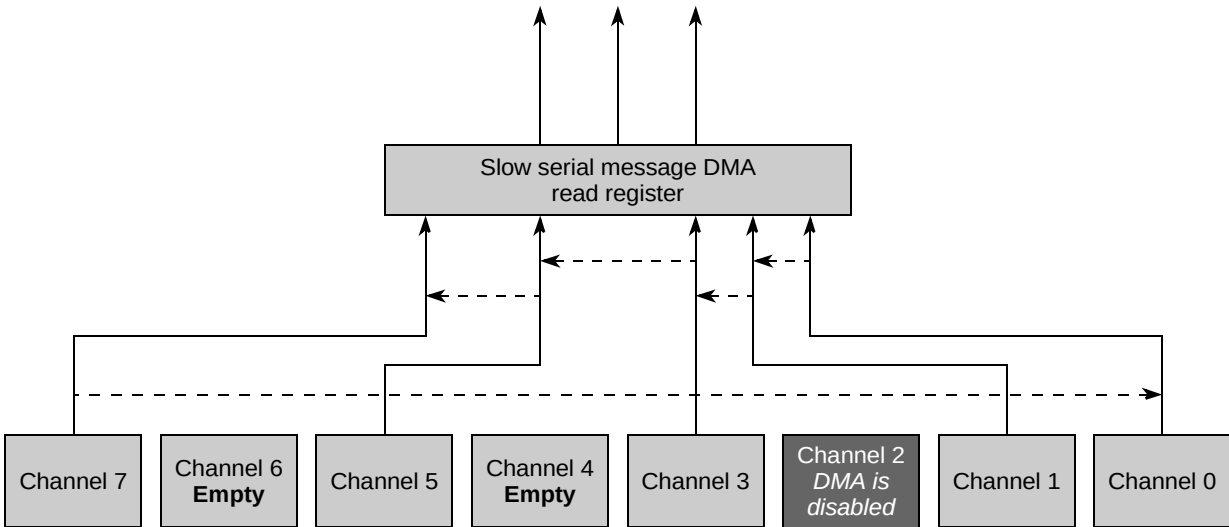
55.4.2.2 DMA request for Slow Serial Message reading

A separate DMA request (different from DMA request for Fast Message) is generated for all slow serial messages received on all channels. The data flow will happen as follows:

- Request will be asserted when any of the DMA enabled channels has successfully received one complete serial message (short or enhanced) and kept asserted till all messages have been read from their respective memory-mapped registers
- Similar to DMA read for Fast Messages, data for slow serial messages will be read from DMA Slow Serial Message Read register set in memory map, the register set comprising of:
 - [DMA Slow Serial Message Bit3 Read Register \(DMA_SMSG_BIT3\)](#)
 - [DMA Slow Serial Message Bit2 Read Register \(DMA_SMSG_BIT2\)](#)
 - [DMA Slow Serial Message Time Stamp Read Register \(DMA_SMSG_TS\)](#)
- A fixed round robin sequencing logic will be running and it will update the DMA Slow Serial Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 DMA read accesses to the above mentioned registers)
- The channel number and time stamp will also be appended for every message
- A message type bit (TYPE) will also be appended to distinguish between short and enhanced serial messages
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- Since serial messages receive rate is very slow, user software should ensure there is no overflow
- The fixed round robin sequence logic for DMA read is the same as that used for Fast Messages (see [Section 55.4.2.1, DMA request for Fast Message reading](#))

[Figure 1285](#) shows the sequencing of channels for serial messages to be read when DMA read is done on DMA Slow Serial Message Read register set.

Figure 1285. Slow Serial Message DMA read logic



Note: The arrows in the figure indicate next DMA transfer. One DMA Read Register set (3 registers) will be read in

The formats in which the 3 types of Serial Messages will be stored in the buffers are shown in figures below.

Figure 1286. Enhanced Serial Message with 8-bit ID field

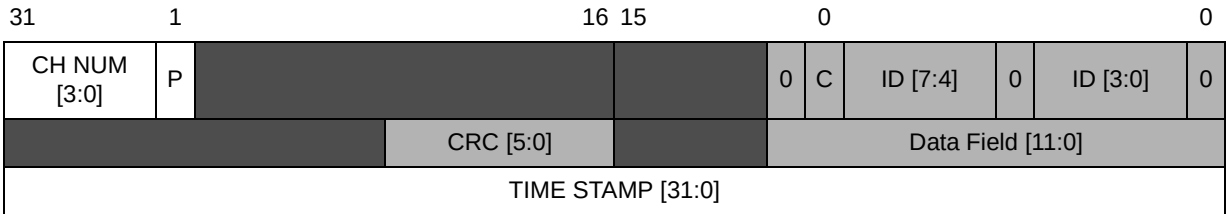


Figure 1287. Enhanced Serial Message with 4-bit ID field

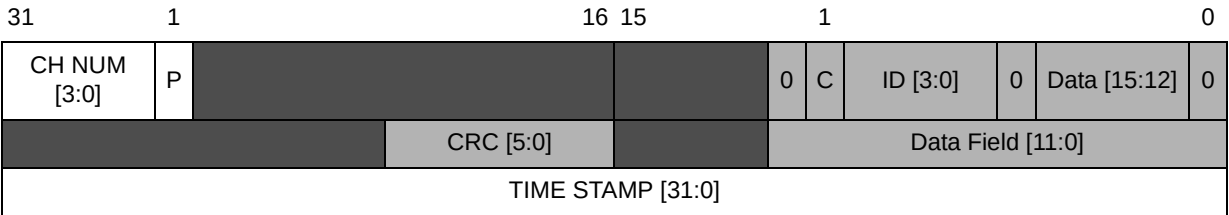
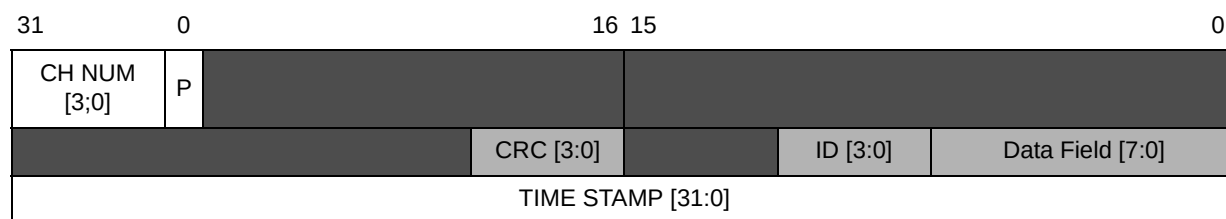


Figure 1286 and Figure 1287 shows the enhanced serial messages. Bit C (shown in figure) or bit CFG (as shown in [DMA Slow Serial Message Bit3 Read Register \(DMA_SMSG_BIT3\)](#) description) defines the type of enhanced packet and bit P (shown in figure) or bit TYPE (shown in [DMA Slow Serial Message Bit3 Read Register \(DMA_SMSG_BIT3\)](#) description) defines whether it is short serial or enhanced serial message. CRC, ID and Data fields are stored in format as received and shown in SAE specification with just one exception that CRC is shifted to align with 16-bit boundary.

Figure 1288 shows the short serial message. Field locations have been placed in such a way to allow a common register to read all the slow serial messages and also match the bit positions in actual message defined by SAE Specification.

Figure 1288. Short serial message



Again, three 32-bit accesses to be made by DMA to get one complete message with time stamp.

55.4.3 Message reading via interrupts

There is one interrupt request per channel for a Fast message received and one for a serial message received. Each channel is individually configurable to enable or disable interrupt on reception of Fast or Slow Serial messages separately. In case of interrupt request of Fast Message, an interrupt will be asserted when any of the channels has received a complete Fast Message without errors and that channel is not configured for DMA Reads. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing all channels that are enabled to give out messages via interrupts through the Register Interface.

In response to these interrupts, the CPU is expected to read interrupt status register for fast or Slow Messages (depending on which interrupt was asserted) and store it in system memory. Using this status register value, the interrupt routine can decide from which channels the message is to be read. For instance if bit corresponding to channel 0 is set then routine should read from the address location corresponding to Channel 0 Fast Message Read registers and move onto the next asserted bit. The interrupt routine should scan through all the bits of this stored interrupt status register value and read corresponding channel's fast or Slow Message registers. In this manner all available messages can be read. After scanning through all the bits of status register stored in system memory, routine can read again the message ready interrupt status register to find out if any other channel has received in the mean time. This will save another interrupt routine time mainly from stacking/unstacking delay.

It is assumed that the interrupt controller will be able to respond fast enough to avoid any overflow as only one message per channel will be stored and the system bus clock frequency is sufficient to avoid this.

55.4.3.1 Suggested software interrupt service routine steps

These are suggested step to have an interrupt routine that can read all messages quickly and avoid any underflow/overflow. Actual implementations can vary.

When separate Fast Message and Slow Serial Message Interrupts are used:

The Fast Message Interrupt should have higher priority than Slow Serial Message Interrupt. The suggested ISR steps for a Fast Message Interrupts can be

- Read the Fast Message Ready Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register set (3 registers). Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Fast Message Status Registers again to ensure no new message has arrived during the ISR.
- If there are no bits set, exit the ISR, else repeat this process.

The ISR steps for Slow Serial Message Interrupt can be same as for Fast Message ISR.

When common Fast Message and Slow Serial Message Interrupts are used:

The suggested ISR steps can be

- Read the Fast Message Status Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Slow Serial Message Status Register into a new variable
- Shift out bits starting from MSB from this new variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Slow Serial Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, software can again check both Fast and Slow Serial Message Ready register to ready any message that might have been received while the routine was being serviced, or can choose to exit the routine.

55.4.4 Overflow behavior

This section describes the overflow behavior.

55.4.4.1 Fast messages buffers

Overflow in Fast Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, the newly received message overwrite the previously stored message and the overflow status bit is set. However, if overflow asserts while CPU/DMA is reading that channel (i.e. all 3 read registers have not yet been read), then that message is not overwritten. While messages are being overwritten, the slow message reception is not hampered.

55.4.4.2 Slow message buffers

Overflow in Slow Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, further reception of slow messages is halted until the buffers are read out by CPU or DMA. Since it is less likely that CPU or DMA do not read the read registers for 32 or 36 fast message length duration, an overflow in slow messages would not be as frequent as in fast messages.

55.4.5 Adjustment for variation in sensor (Tx) clock

The clock used by the Channel Receiver (referred to as Receiver Clock, or Rx_CLK) to sample the message nibbles is generated from a High Frequency Receiver Clock or Protocol Clock. This high frequency receiver clock is divided using a prescaler counter which generates a tick when it rollovers on reaching a count equal to the Prescaler Value programmed in the channel's clock control register. The receive clock is generated separately for each channel. By varying this Prescaler Factor, we can control the period of the receive clock and make it close to the Transmitter clock used by the sensor for the particular channel.

After reset and once channel is enabled, the compensated prescaler value is set to Zero and the user software programs the Prescaler Value to obtain the desired receiver clock frequency for the particular channel. Thus, the receiver starts working by assuming that the transmitter is sending messages on the frequency that is programmed in its channel clock control register. The channel's input after filtering is continuously measured by the Prescaler Counter. The compensation logic checks each pulse length and determines the calibration pulse from them. When a calibration pulse is detected, the compensated prescaler value is computed and stored in the Channel's Clock Control Register ([Channel 'n' Clock Control Register \(n = 0 to \(CH-1\)\) \(CHn_CLK_CTRL\)](#)). Software can use this value to determine the variation in Tx Clock.

Note: A pause pulse may incidentally be detected as a synchronization/calibration pulse during start up after reset. The correction value would be computed on the pause pulse too but since the next synchronization pulse would follow, the logic would again detect the falling edge on the synchronization pulse and the correction value would now be correctly computed on the correct synchronization pulse.

The Clock Compensation or Correction happens for every messages that is received and the compensated prescaler value remains constant for that message.

55.4.5.1 Adjustment for nibble length variation

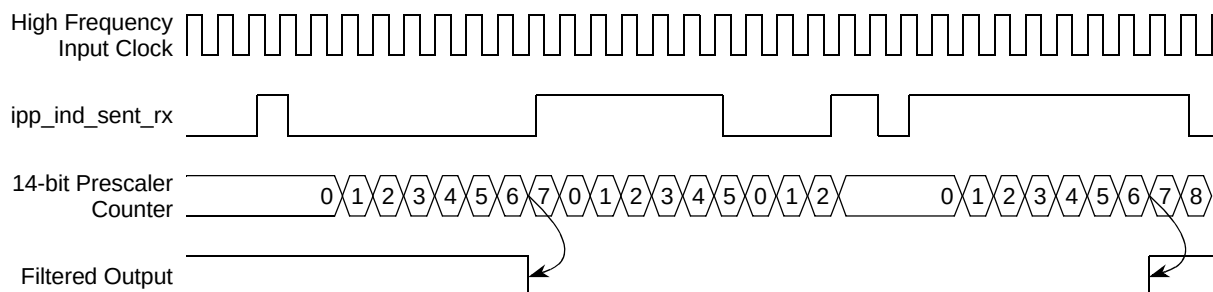
Section 6.2 of SAE Specifications specifies the maximum allowable limits for clock drift and jitter for the transmitter and receiver separately. This can cause the length of nibbles to vary and can cause incorrect sampling of data nibbles. This variation is adjusted by the SENT Receiver when nibbles are being sampled.

55.4.6 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs

in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflow occurs, the new pin value (i.e. sensor input) is validated and latched in the filtered output flop (also running on high frequency receiver clock). If the opposite edge appears on the sensor input before validation (i.e. overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register (*Channel 'n' Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*). When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register (*Channel 'n' Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*). A timing diagram of the input filter is shown in *Figure 1289*.

Figure 1289. Input programmable filter timing diagram



55.4.7 Receiver diagnostics

Most of the receiver diagnostics are controlled by the Fast Message State Machine. The following checks are conducted by the receiver in each channel

- Calibration pulse length < 56 clock ticks – 25% or > 56 clock ticks + 25%
- Not the expected number of falling edges between calibration pulses. (Message length is pre-defined by each sensor device and programmed by user for each channel)
- Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or XOR-based.
- Any nibble data values measured as < 0 or > 15
- Successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used.
- For messages with pause pulse, an additional diagnostic is done; however, this diagnostic will not fail any message (to reduce latency in message reception) but the failure of this diagnostic will be indicated in the channel's status register.

On detection of any of the above errors, the receiver rejects the current packet until a new calibration pulse is detected. Occurrence of an error is flagged in the respective channel's status register (*Channel 'n' Status Register (n = 0 to (CH-1)) (CHn_STATUS)*). If error interrupt is enabled, an error interrupt will be generated. There is a single error interrupt generated for all errors, so the CPU must read the status register to find out which error occurred. Once error condition is flagged, subsequent errors (except NUM_EDGES_ERR)

are automatically masked out until a valid calibration pulse is detected, after which all checks are automatically enabled. This is done to avoid multiple assertions of the interrupt to CPU in case the channel become unstable and send wrong data continuously.

55.4.7.1 Calibration pulse length check

The calibration pulse length check happens in conjunction with Clock Compensation Logic since current calibration pulse is detected in this block. The Clock Compensation Logic detects a calibration pulse of length between 42 and 70 tick counts. This check will assert the CAL_LEN_ERR status bit on failure.

55.4.7.2 Successive calibration pulse check

The successive calibration pulse check checks if successive calibration pulses differ by $> +1.5625\%$ ($1/64$) or $< -1.5625\%$. The accuracy of this check will depend on the frequency of the protocol clock used. Specifically, the higher the frequency, the closer the check will be to 1.5625% . So if protocol clock is T_{HF_CLK} and channel tick period is T_{TX_CLK} then the check will be of $1.5625\% \pm \{100 \times (4 \times T_{HF_CLK}) / (42 \times T_{TX_CLK})\}$. So it won't allow variance in clock of a channel by $1.5625\% - \{100 \times (4 \times T_{HF_CLK}) / (42 \times T_{TX_CLK})\}$. This check will assert the CAL_DIAG_ERR status bit on failure.

When Option 2 of successive calibration pulse check is enabled, the CAL_RESYNC bit will be asserted on every third successive error detected by this check.

55.4.7.3 Not the expected number of edges check

The number of negative edges between two successive calibration pulses is counted and compared with expected number of edges in case of preferred option of successive calibration pulse check. The expected number of nibbles is the sum of following:

- Status and Communication nibble (1 No.)
- Data nibbles (Programmable as per user software)
- CRC nibble (1 No.)
- Pause Pulse, if configured (1 No.)

This check ("Not the expected number of edges check") will assert the NUM_EDGES_ERR status bit on failure. The check does not run when Option 2 (low latency) of successive calibration pulse check is enabled. This check is not masked after any other error is detected so it could be redundant if it is flagged along with CAL_DIAG_ERR and CAL_LEN_ERR. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

55.4.7.4 Nibble value check

As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 ticks:

- Status and Communication nibble
- all Data nibbles
- CRC nibble

The nibble value check will assert the NIB_VAL_ERR status bit on failure. This error can be flagged along with NUM_EDGES_ERR in which case the former can be ignored as the number of pulses in the message is incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. In case, errors are accumulated, the above cannot be deduced.

55.4.7.5 CRC check

The CRC check will compute both 4-bit and 6-bit CRC as requested by State Machine control logic. Two 4-bit CRCs and one 6-bit CRC are supported to compute the CRC on Fast/Slow Serial Message (4-bit CRC) and Enhanced Serial Message (6-bit CRC) respectively. The SENT Receiver Module allows the user to calculate the 4-bit CRC using either the legacy CRC implementation method or the new XOR-based CRC implementation via a control bit in the channel's corresponding control register. Please refer to SAE SENT Specification for details on CRC algorithm.

Asserting the CHn_CONFIG_REG[11] bit, causes the Status and Communication nibble to be included in the 4-bit CRC calculation for Fast Serial Messages. De-asserting this bit enables normal operation as described above.

This check will assert the FMSG_CRC_ERR or SMSG_CRC_ERR status bit on failure, depending on whether the check is run for Fast Message or Slow Message.

The FMSG_CRC_ERR can be flagged along with the NUM_EDGES_ERR, in which case the CRC error can be ignored as the number of pulses in the message was incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

55.4.7.6 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per spec). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

55.4.8 Time stamp logic

A single counter maintains the time stamp for all channels which can be used by the software to determine the relative time of arrival of the messages for each channel. The time stamp counter is a 32-bit counter.

The clock for this counter is derived from the high frequency receiver clock using a Time Stamp Prescaler Counter that divides the high frequency receiver clock to generate the clock of required resolution. The user must program the time stamp prescaler value (in [Global Control Register \(GBL_CTRL\)](#)) in order to generate the time stamp clock. The period of the time stamp clock must be less than that of the fastest receiver channel being used, or it can be set at some fixed value determined by application, such as 1 μ s. For example, if the high frequency receiver clock is 60 MHz, the required prescaler value to generate a time stamp clock of 1 μ s resolution is 60.

Note: The rollover of time stamp values should be determined in user software, by comparing the time stamp values in previous and current messages that are read out. User software should adjust the time stamp values in these messages accordingly, after reading them.

The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer. However, if the current message is found to be in error, the message buffer is discarded. Thus, the time stamp will be available to user software for only those messages which do not have any error.

The time stamp for a serial message is sampled at the end of the message i.e. when the serial message is completely received and when the message is found to have no errors. the time stamp is stored in the message read buffer to be read by the user software. Again, for messages with error the message and time stamp are discarded.

55.4.8.1 Limitation

Since the same counter is used for all channels, it may happen that more than 1 message is received within the same time stamp across different channels. Hence the software would not be able to determine the sequence of arrival of messages which have the same time stamp value.

55.4.9 Bus Idle Diagnostic

This diagnostic checks the connected sensors for inactivity. SENT module can indicate if the idle period on a particular channel has crossed the period defined by the programmed value in Bus Idle Count (in [Channel 'n' Configuration Register \(n = 0 to \(CH-1\)\) \(CHn_CONFIG\)](#)). Status bit (bus_idle in [Channel 'n' Status Register \(n = 0 to \(CH-1\)\) \(CHn_STATUS\)](#)) will be set when idle period crosses the allowed value. CPU should write 1 to this clear this bit once it is read.

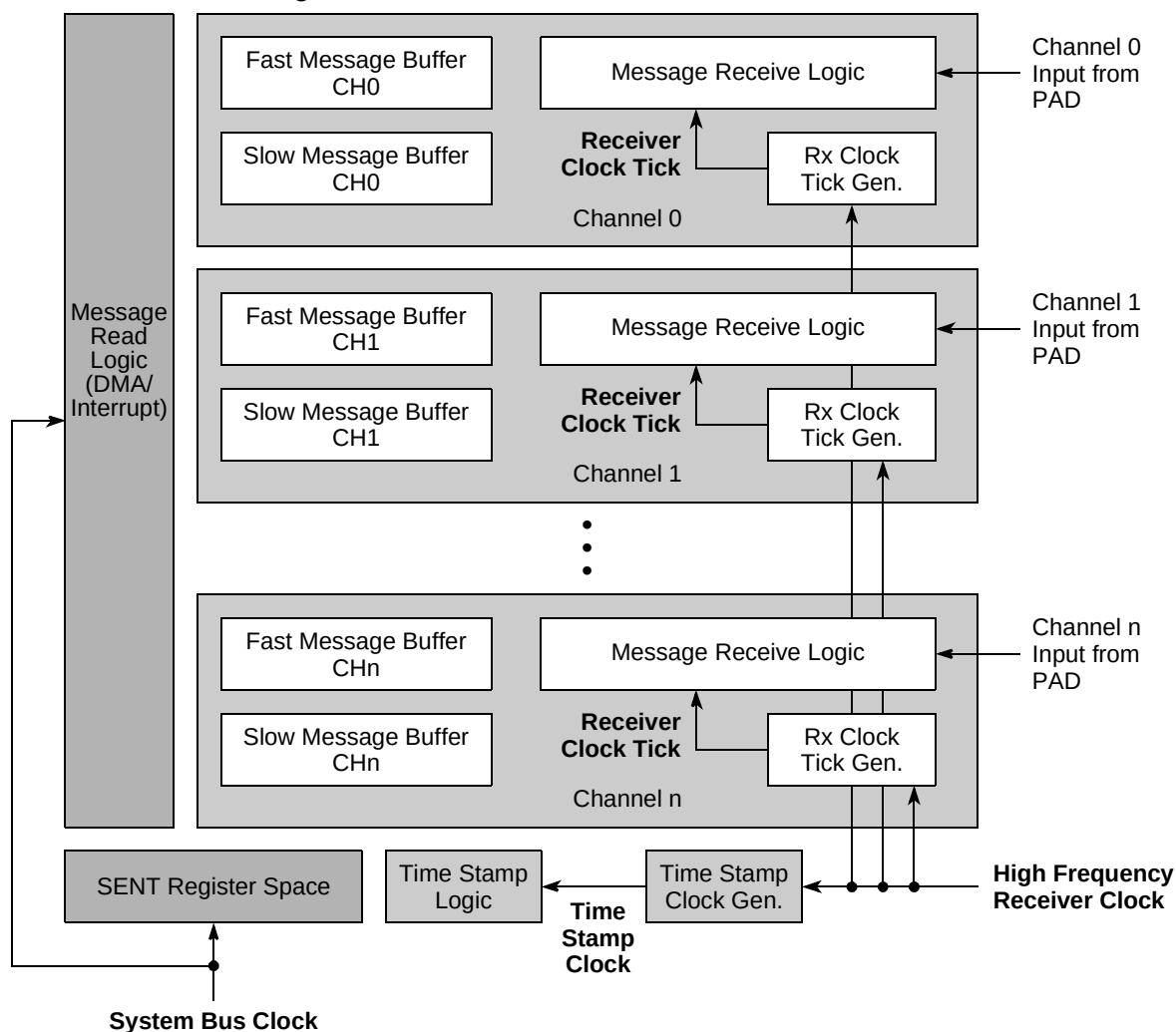
55.5 Clocks and resets

The SENT Receiver works on two clocks. First is the System Bus Clock that is used to program the registers and read messages via DMA or Interrupt from the module's Register Interface. Second is the High Frequency Receiver Clock or Protocol Clock that is used for accurate message receiving operations. There is a single asynchronous reset which is the system asynchronous reset and one internal reset for each of the channel logic.

55.5.1 Clocking strategy

[Figure 1290](#) shows the clock domain in which each module is functioning.

Figure 1290. SENT Receiver module clock domains



55.5.2 System bus clock requirements

Since all the received messages are read on the System Bus Clock it is necessary that both the interrupt service routine and the DMA controller must be able to access all channels with messages ready to be read within a time to prevent overflow in any channel. The minimum time available to the CPU or DMA is the case when the smallest Fast Message is received back to back on all channels while working on the smallest receive clock tick period.

Now, the smallest Fast Message size with valid checksum as per SAE Specifications is 154 ticks. At 3 μ s receive clock tick with a -20% variation in the transmitter clock tick, the effective receiver clock tick is 2.4 μ s (after compensation). This gives us the time in which the minimum size Fast Message is received in any channel as $154 \times 2.4 = 369.6 \mu$ s. Now since messages are arriving in parallel, 369.6 μ s is the time available for CPU or DMA to access all channels to prevent an overflow in the channel that received the very first message.

Thus, if assume that ' N_{CHNL} ' channels are being used and all channels have messages ready to be read together from either DMA or CPU then the time available to read a message in one channel will be $369.6/N_{CHNL} \mu$ s or $\sim 370/N_{CHNL} \mu$ s. Since the CPU or the

DMA needs to make three 32-bit accesses in order to read a complete message and taking the time required for one 32-bit read access as ' T_{RD} ' (assuming it to be same for DMA and CPU); we can determine the minimum frequency for the System Bus Clock as:

For CPU,

$$\text{Equation 82} \quad 3 \times T_{RD} + T_{ISR} + T_{WAKUP} < 370/N_{CHNL}$$

where T_{ISR} is the overhead in initiating an ISR and T_{WAKUP} is the time to wakeup from low power mode

For DMA,

$$\text{Equation 83} \quad 3 \times T_{RD} + T_{WAKUP} < 370/N_{CHNL}$$

Replacing $T_{RD} = N_{RD} \times T_{SYS_CLK}$ and $T_{ISR} = N_{ISR} \times T_{SYS_CLK}$, where N_{RD} is the number of System Bus Clock cycles in one 32-bit read access, N_{ISR} is the number of System Bus Clock Cycles in initiating an interrupt service routine and T_{SYS_CLK} is the period of the same clock, we get

For CPU,

$$\text{Equation 84} \quad (3 \times N_{RD} + N_{ISR}) \times T_{SYS_CLK} + T_{WAKUP} < 370/N_{CHNL}$$

(all units is μs)

For DMA,

$$\text{Equation 85} \quad 3 \times N_{RD} \times T_{SYS_CLK} + T_{WAKUP} < 370/N_{CHNL}$$

(all units is μs)

55.5.3 High frequency receiver clock (protocol clock) requirements

For accurate receive operations, the protocol is required to satisfy a minimum frequency defined as per formula below.

$$\text{Equation 86} \quad F_{HF_CLK} > 332 / (3 \times T_{TX_CLK})$$

where F_{HF_CLK} is the High Frequency Receiver Clock or Protocol Clock in MHz and T_{TX_CLK} is the Transmitter or Sensor Clock Frequency in μs .

Note: T_{TX_CLK} in the above equation should be chosen based on the fastest sensor clock among all channels in the device.

56 LINFlexD

56.1 Introduction

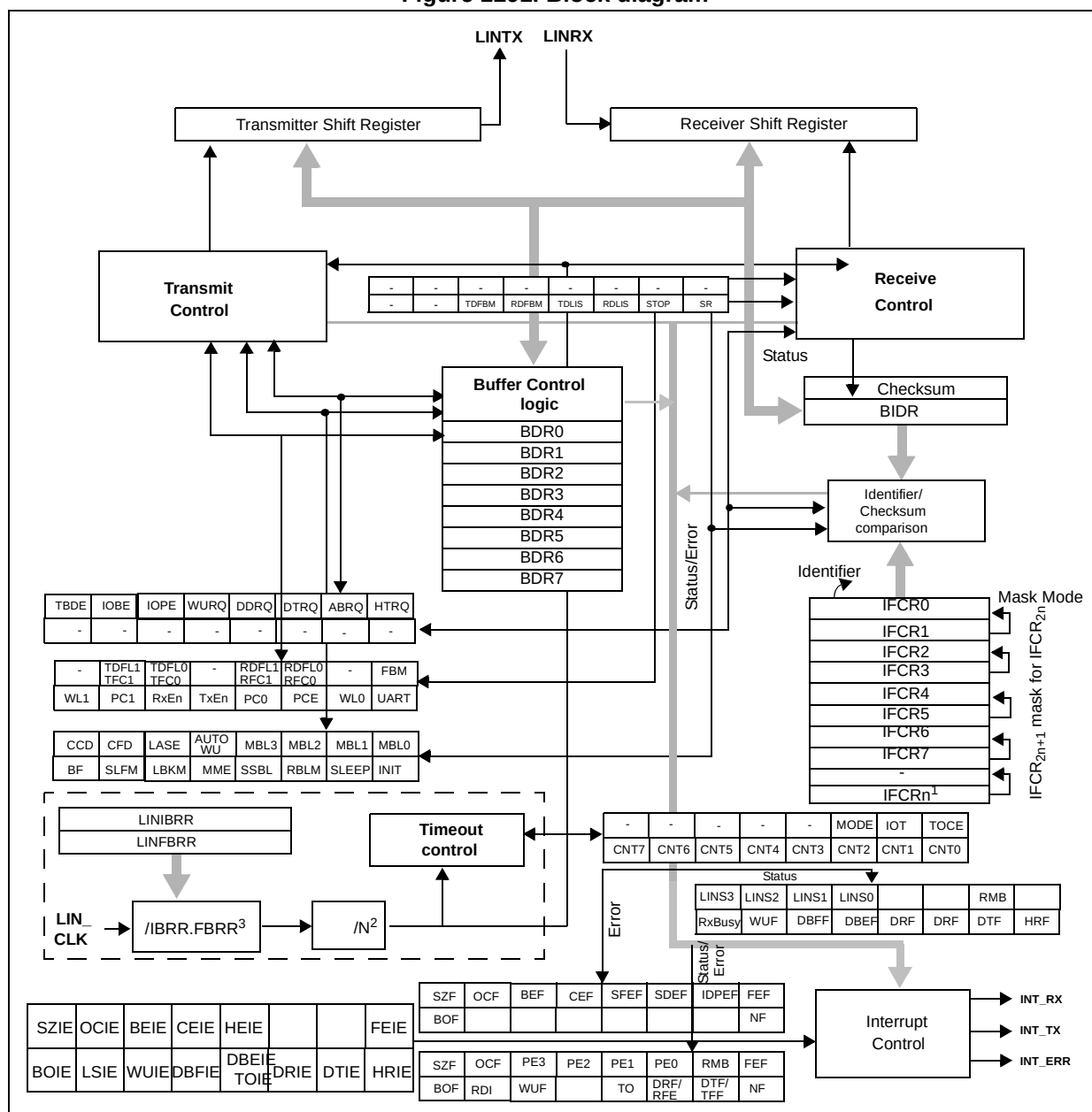
The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, and 2.1. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/UART operating mode.

Figure 1291. Block diagram



1. The value of n depends on the no_of_filters. Refer to the device configuration chapter to see the number of filters used in this device.
2. For LIN mode N = 16 and N is programmable by user in UART mode.
3. FBRR is not usable when reduced over-sampling is enabled in UART mode.

56.2 Main features

The LINFlexD controller can operate in several modes, each of which has a distinct set of features which are described in the following sections. In addition, the LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
 - Initialization
 - Normal
 - Sleep
- Test Mode: Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

56.2.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, and 2.1
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection
- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
 - Autonomous header handling
 - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation (The relationship “ $\text{LIN_CLK} > \text{PBRIDGE_CLK} \geq 1/3 * \text{LIN_CLK}$ ” should be maintained.)

56.2.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBR registers — see [Section 56.3.2.8, Baud rate generation](#)
- Separate clock for baud rate calculation (The relationship “ $LIN_CLK > PBRIDGE_CLK \geq 1/3 * LIN_CLK$ ” should be maintained)
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management
- The maximum baud rate achievable is 25 Mbit/s.
- For bit rate ≤ 6.25 Mbit/s
 - Sixteen times oversampling
 - 3:1 majority voting
- For 6.25 Mbit/s $<$ bit rate ≤ 12.5 Mbit/s
 - Reduced over sampling programmable by the user
 - 3:1 majority voting for reduced over sampling of 8
- For 12.5 Mbit/s $<$ bit rate ≤ 25 Mbit/s
 - Reduced over sampling programmable by the user
 - 1:1 voting for all reduced over sampling of 4, 5 and 6

56.2.3 Standard features

DMA2x bus interface

56.3 Functional description

56.3.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task — all other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

56.3.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

Figure 1292. Frames

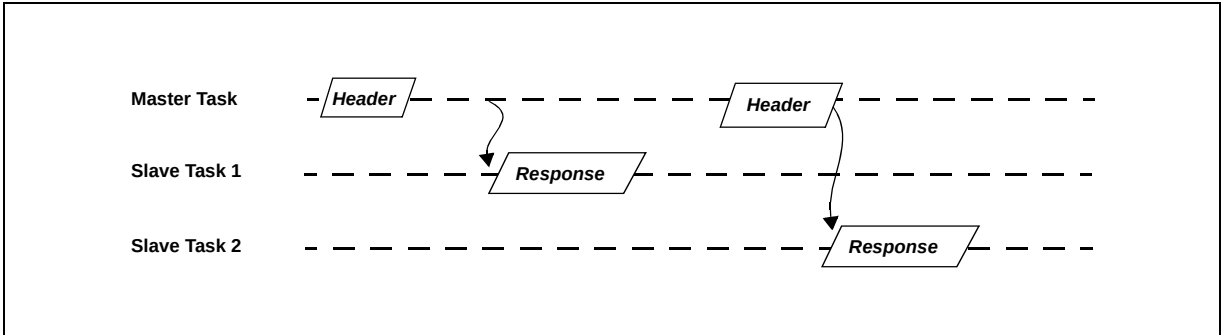
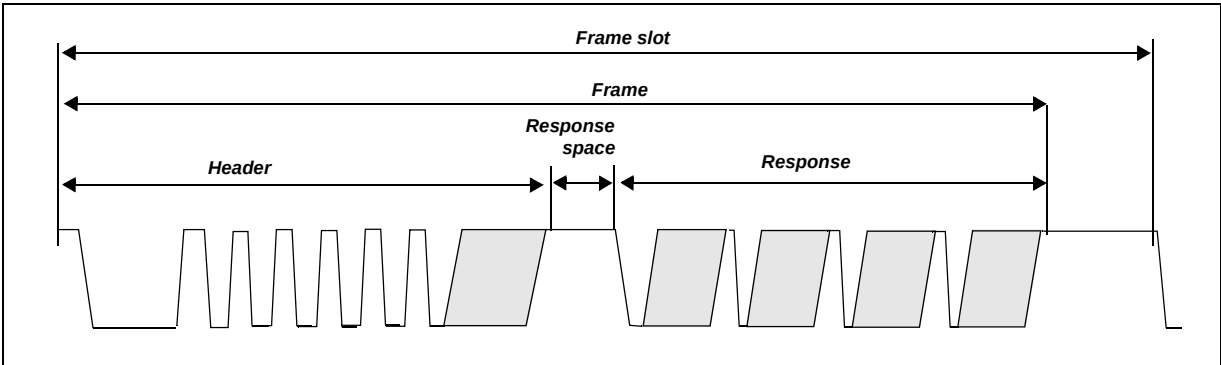


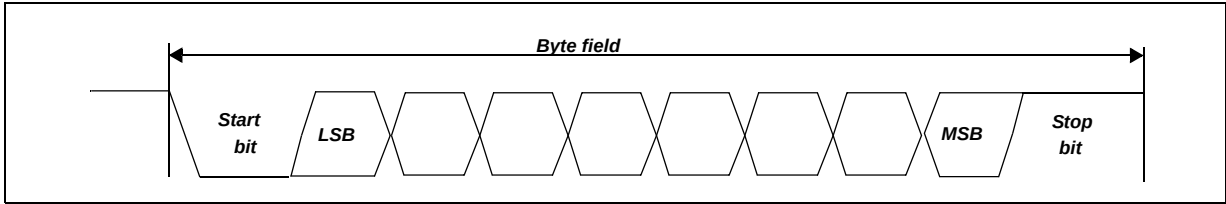
Figure 1293. Structure of LIN frame



56.3.1.2 Data field

Each byte is transmitted as shown in [Figure 1294](#). The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

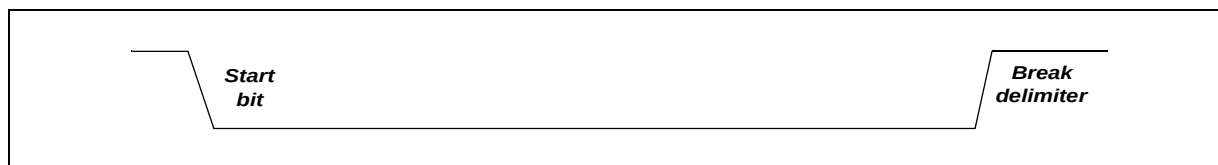
Figure 1294. Structure of byte field



56.3.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in [Figure 1295](#). The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

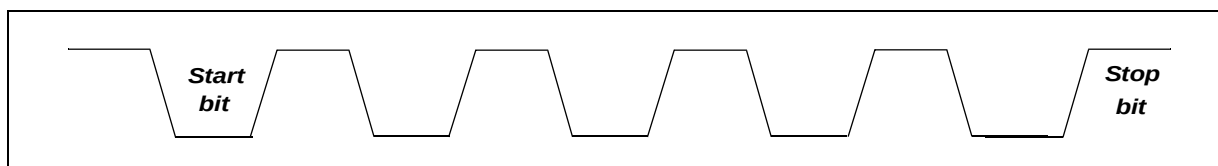
Figure 1295. Break field



56.3.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

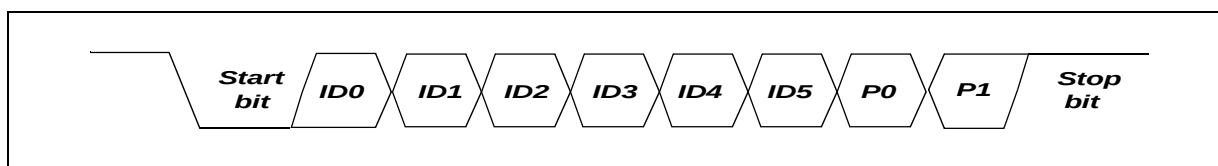
Figure 1296. Sync byte field



56.3.1.5 Identifier

The Identifier field consists of two sub-fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

Figure 1297. Identifier field



$$P0 = ID0 \text{ XOR } ID1 \text{ XOR } ID2 \text{ XOR } ID4$$

$$P1 = \text{NOT } (ID1 \text{ XOR } ID3 \text{ XOR } ID4 \text{ XOR } ID5)$$

56.3.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum, and it is used for communication with LIN 2.0 slaves.

56.3.2 LINFlexD features

56.3.2.1 Operating modes

The LINFlexD has three operating modes: **Initialization**, **Normal**, and **Sleep** modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINC1R1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller, software must:

1. set up the baud rate registers
2. reset the UART bit
3. select the mode (Master or Slave)
4. configure checksum control bits
5. initialize the identifier list (Slave mode)

Normal mode (NM)

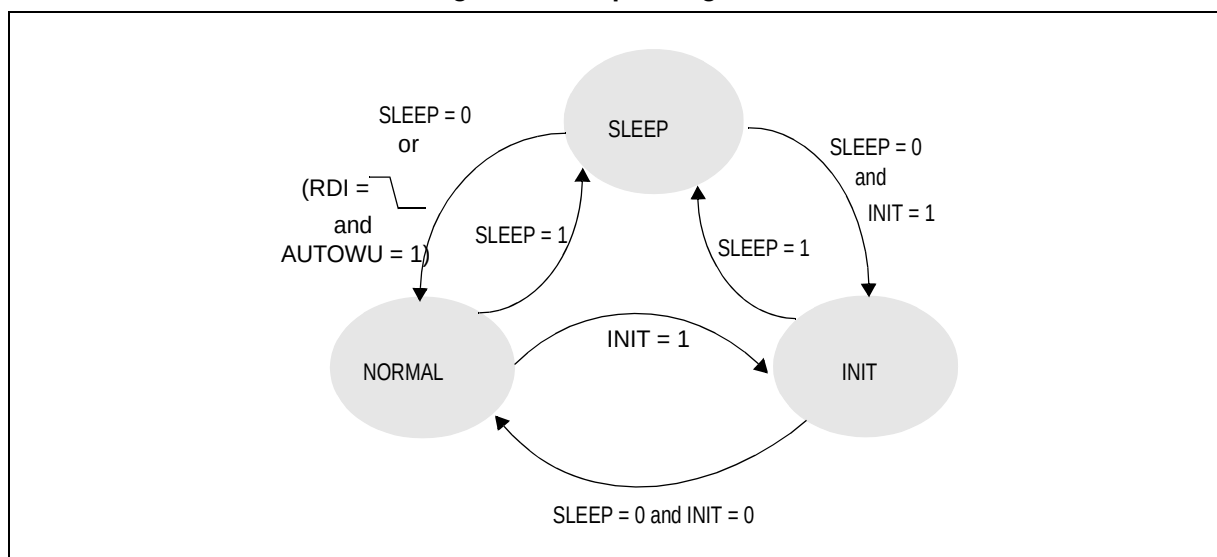
Once software has completed initialization of the LINFlexD controller, it can enter Normal mode by clearing the INIT bit.

Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINC1. In this mode the LINFlexD clock is stopped. LINFlexD can be woken from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 μ s on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to [Section 56.3.2.10, Wakeup management](#).

Figure 1298. Operating modes

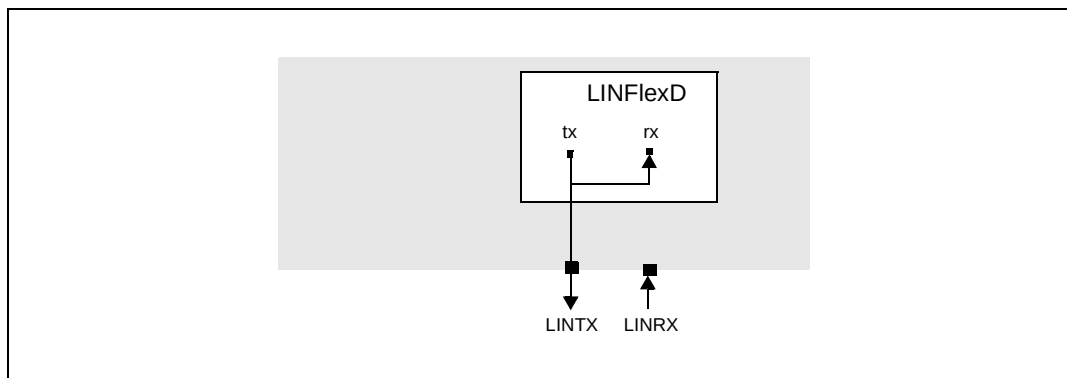


56.3.2.2 Test mode

Loop Back mode

This mode is entered by setting the LBKM bit in LINC1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.



56.3.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits until the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDR[0:7]. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling). It is possible to handle frames with a response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, LINSR[DBEF] is set once the first eight bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset. Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling). It is possible to handle frames with a Response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, the LINSR (DBFF) bit is set once the first eight bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

56.3.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. read the received ID in the BIDR register
2. fill the BDR[0:x] register
3. program the CCS and DIR bits in the BIDR register
4. specify the data field length DFL[5:0] bits in the BIDR register
5. trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7]. The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software, you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

56.3.2.5 Errors

56.3.2.5.1 Header error

A header error is an error during header reception. The types are:

- Sync Del error (SDEF)
- Sync field error (SFEF)
- Identifier Parity error (IDPEF)

1. Sync Delimiter too short

The delimiter should be 1 for at least one bit time, otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if the HEIE bit of LINEIER is set.

2. Inconsistent Sync field

This error condition is monitored differently depending on whether autosynchronization is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the inconsistent Sync Field informs:

- that the deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators,
or
- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator). Two checks are performed in parallel:

1. The first check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

If there is an inconsistent Sync Field error, it means that:

$$D > 14.0625\%$$

If there is no error, it means that:

$$D < 14.84375\%$$

If $14.0625\% < D < 14.84375\%$, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.

2. The second check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = 1)

In this case the Sync character is received as a normal character. If the received character is 0x55, then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

3. Identifier Parity error

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

$$P0 = ID0 \text{ XOR } ID1 \text{ XOR } ID2 \text{ XOR } ID4$$

$$P1 = \text{NOT } (ID1 \text{ XOR } ID3 \text{ XOR } ID4 \text{ XOR } ID5)$$

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

56.3.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CLK cycles.

Example 17

1-bit time at 20 Kbit/s = 50 μ s

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) – (6 LIN_CLK cycles) = 49.925 μ s

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note: *If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.*

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In

this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.

56.3.2.5.3 Framing error

This error is flagged when a dominant state is sampled on the stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if the FEIE bit is set in LINEIER.

The byte that caused framing error is also shifted to the buffer but DRF or DBFF is never set in this case.

56.3.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINEIER.

56.3.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of a slave, if the buffer is not released (RMB is not reset) before reception of the next Identifier and if RBLM is set, then the ID along with the data is discarded.

56.3.2.5.6 Timeout error

Figure 1299. Incomplete response (for example, missing checksum)

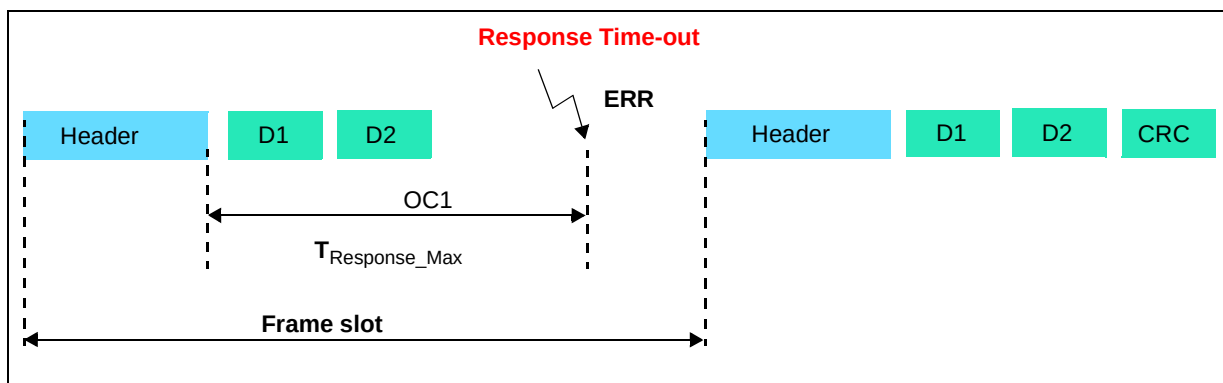
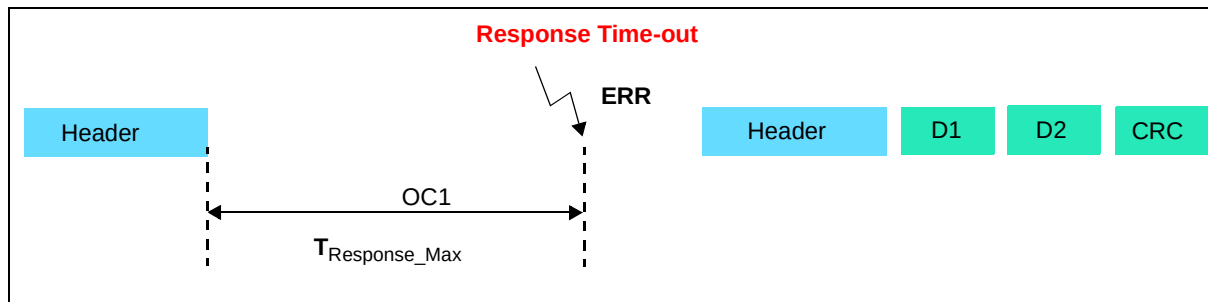


Figure 1300. No response



Response timeout mechanism

- Master mode
 - ocr_l is loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = $1.4 \times ((DFL + 2) \times 10 \text{ bit time})$).
 - The change with respect to earlier implementation is that this loading takes place at the end of the ID field (and not at the beginning of the data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further ocr_l update is required.
 - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the ocr_l value (no change compared to earlier implementation).
 - In case of an incomplete response, timeout also occurs when the counter reaches the ocr_l value (no change compared to earlier implementation).
- Slave mode
 - **Case 1:** The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, ocr_l can be loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = $1.4 \times ((DFL + 2) \times 10 \text{ bit time})$).
 - **Case 2:** The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the previous strategy cannot be used as the DFL value is not yet available. The earlier implementation loaded the maximum value = $1.4 \times (9 \times 10 \text{ bit time})$ and then updated it according to DFL after the first data byte reception. This was not acceptable if no data byte was received at all, as timeout expired too late. The proposed implementation is the following:
 - At the end of the ID, ocr_l is loaded with 36 (maximum possible response space)
 - At the end of the first_data_byte it is reloaded again according to DFL
 - Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
 - no error on the bus and timing is correct (nominal header length)
 - an error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error)

Therefore there is no meaning of header timeout in master mode, so it is disabled

- Slave mode
 - $\text{header_nominal} = 13 + 2 + 10 + 10 = 35 \text{ Tbit}$
 - $\text{header_max} = 1.4 \times \text{Header_nominal} = 49 \text{ Tbit}$
 - taking into account a possible 14% clock deviation, header_max seen by LINFlexD is $49 \times 1.14 = 56 \text{ Tbit}$.
 - If the counter starts after 11 Tbit, the HTO value is $56 - 11 = 45 \text{ Tbit}$

The reset value of HTO is 45 and this register can only be programmed in slave mode.

As response space is not included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be set and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

56.3.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8th, 9th, and 10th samples. If any one of these three samples has a value different from the other two, this error is flagged.

When OSR = 8, majority of 2nd, 3rd, and 4th samples are taken into account to determine noise. Noise checking is disabled for all OSR less than eight.

This error is flagged when there is noise detected in the start bit (see [Section 56.4.2.4, LIN Error Status Register \(LINESR\)](#)).

56.3.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a

filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number $2n$ and $2n+1$ is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/reception equals `no_of_filters`, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR $2n$ or IFCR $2n+1$ (if the corresponding FACT bit is set).

Identifier mask mode

If the number of filters required is more than `no_of_filters`, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number $2n$ is the filter and $2n+1$ acts as a mask for it. The FACT bit for filter $2n+1$ has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

If there is a match of identifier with the m^{th} filter in any mode, then $m+1$ is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only until the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after a complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

Note: *If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In Mask mode, if two filters match with the identifier then the filter having the lower number prevails.*

Figure 1301. Identifier List mode

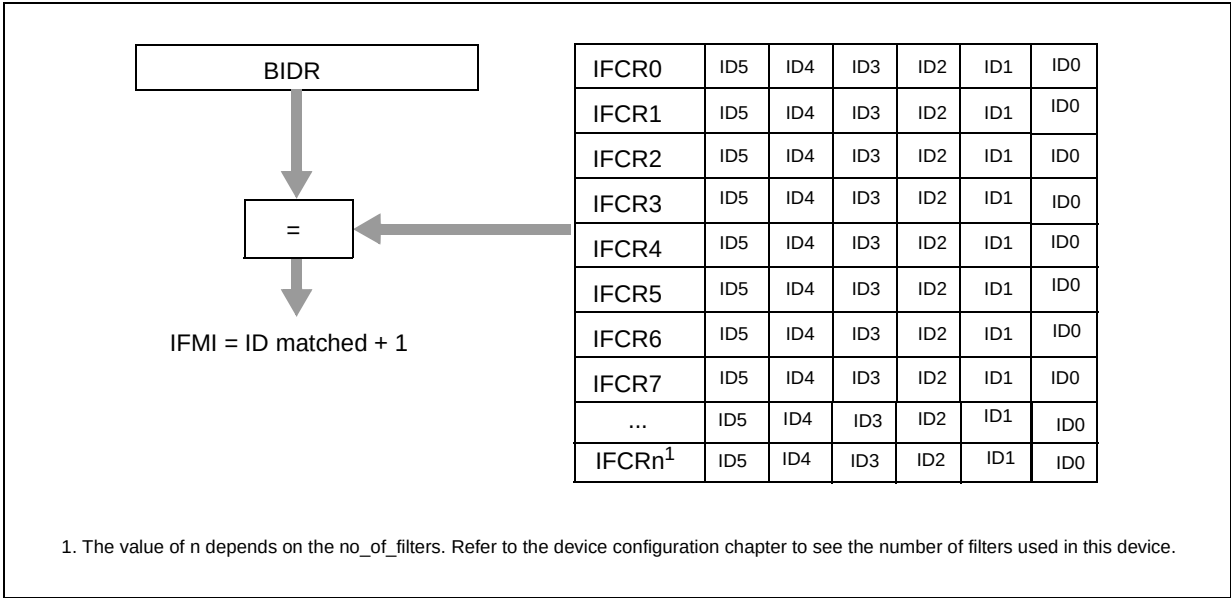
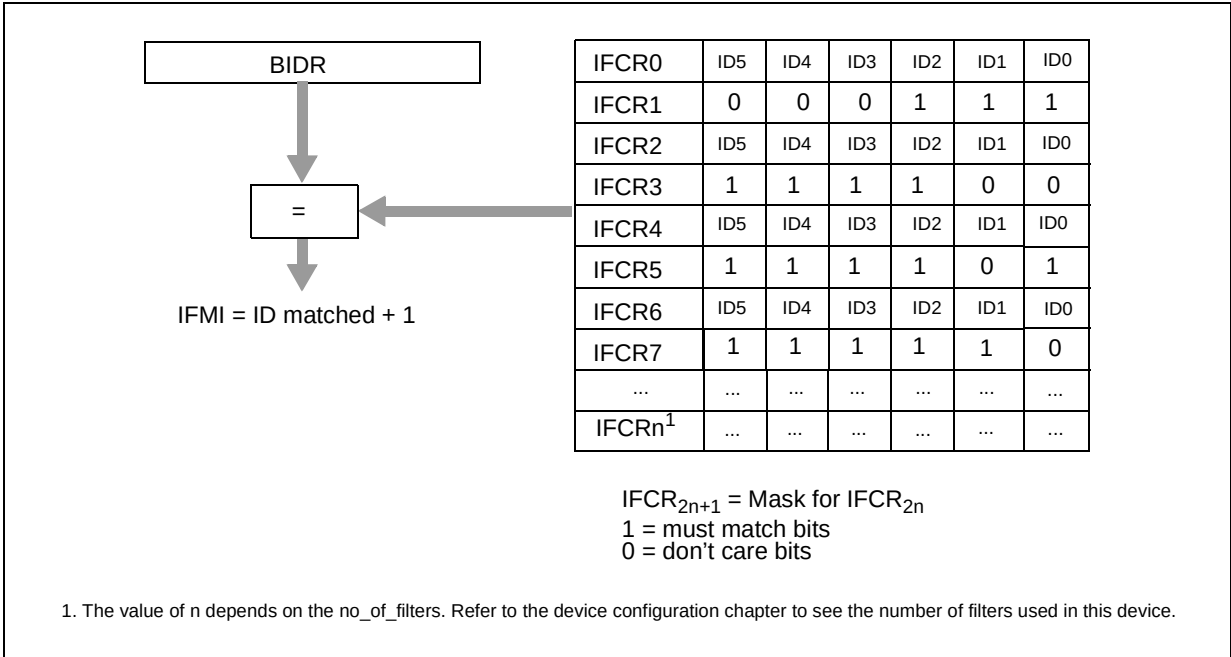


Figure 1302. Identifier Mask mode



56.3.2.7 Start detection and delimiter detection in receiver

There is a 10-bit-shift-register sample register in the receiver that shifts signal data to the next less significant bit on each incoming sample.

Table 1248. 10-bit-shift-register sample register

Sample register bit number										Counter
9	8	7	6	5	4	3	2	1	0	
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x ⁽¹⁾	0	x	0	x	0	Start detection
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	Delimiter detection
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

1. x: Due to sampling during transition of the bit, the bit value might be sampled as a '0' or '1' depending on the rise/fall time. So 'x' refers to an indeterministic value.

56.3.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to [Table 1248](#) for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to '0', then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the "verification samples" which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called "qualification samples" which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to '0' then Noise flag is set.

56.3.2.7.2 Delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to [Table 1248](#) for the status of the sample register (shift register) when count = 6
2. At this point, if RT1(the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to '1', then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the "verification samples" which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called "qualification samples" which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

Figure 1303. Start detection and sampling for over sampling rate = 4 (Case 1)

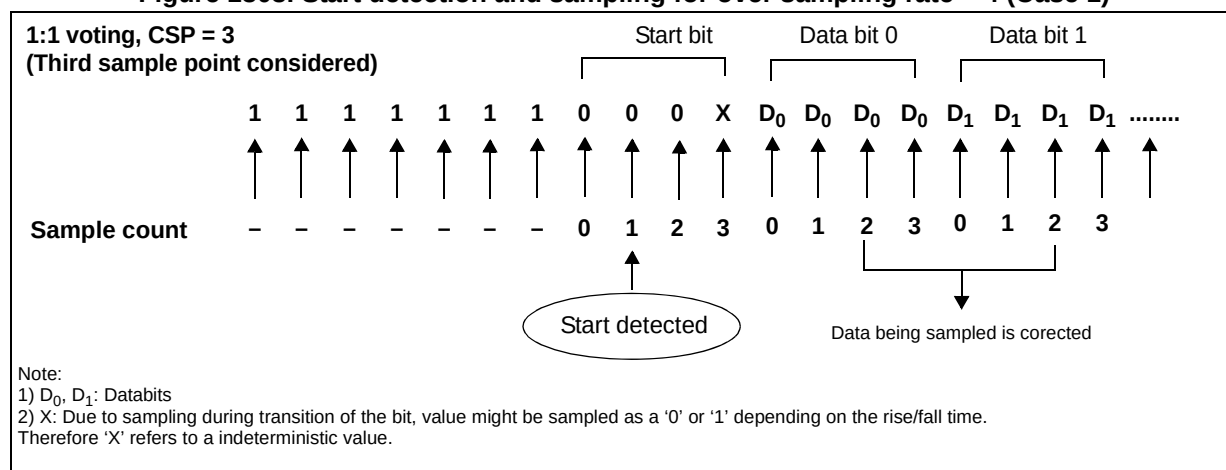
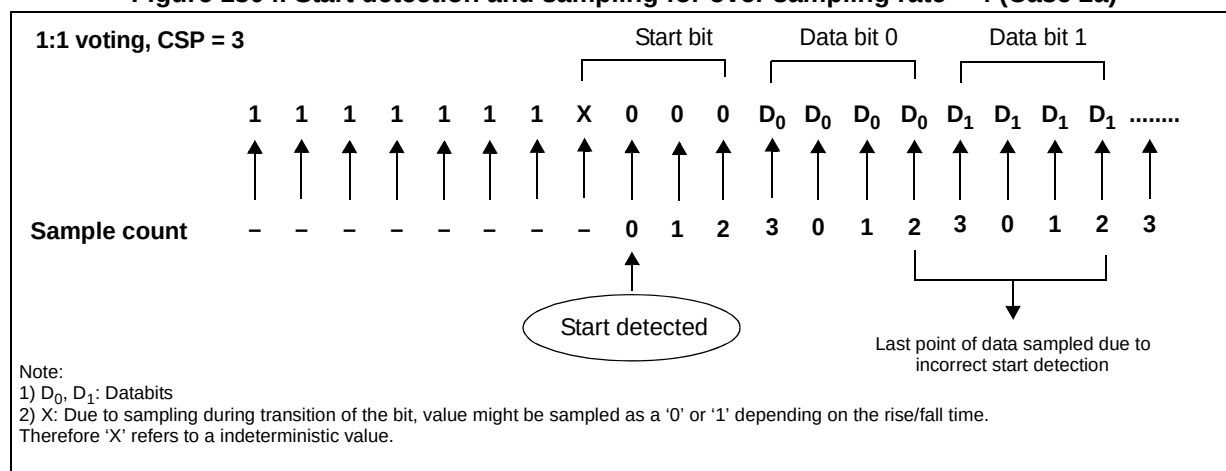


Figure 1304. Start detection and sampling for over sampling rate = 4 (Case 2a)



The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

To sample at the correct point therefore, CSP has to be changed to two; resulting in the following situation:

Figure 1305. Start detection and sampling for over sampling rate = 4 (Case 2b)

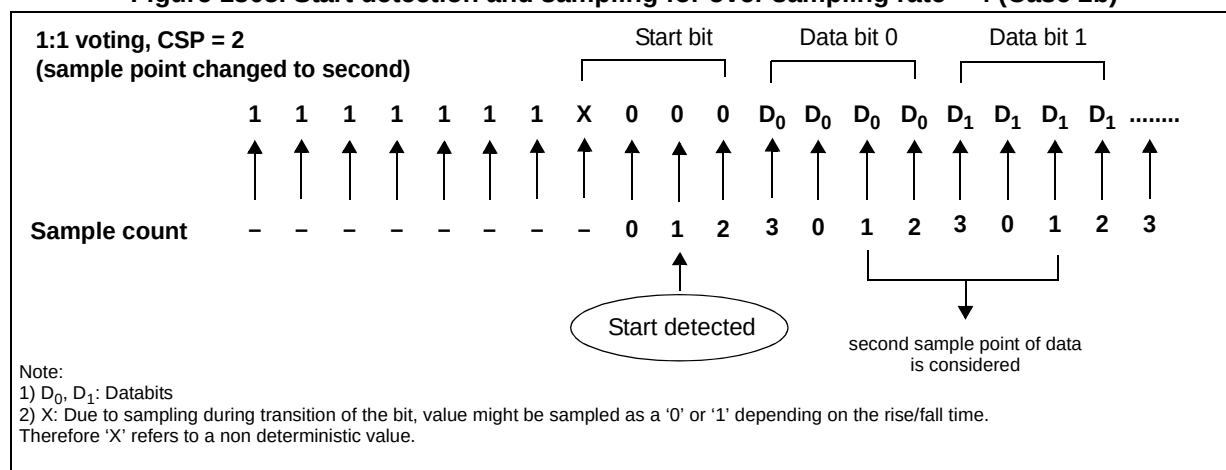


Figure 1306. Start detection and sampling for over sampling rate = 4 (Case 3)

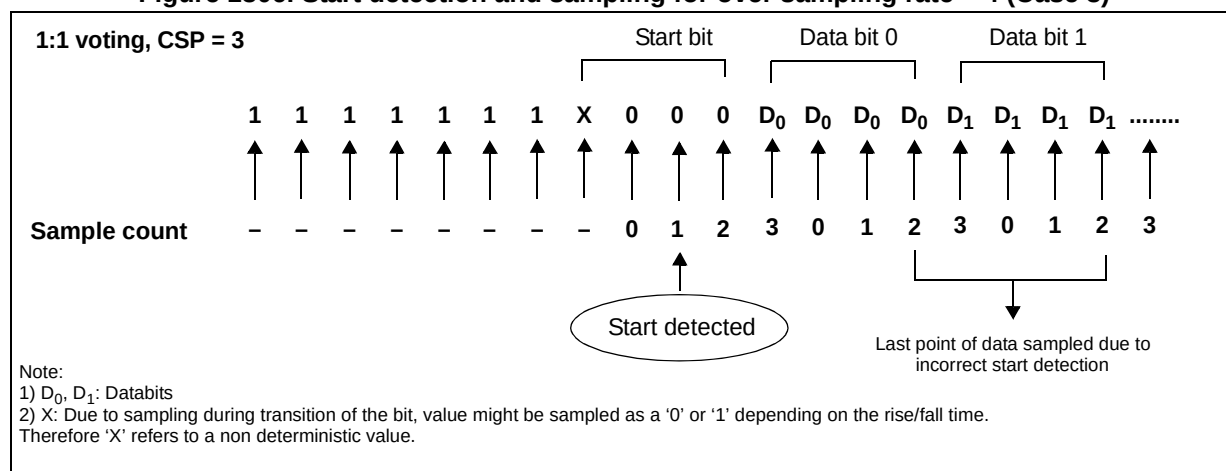


Figure 1307. Start detection and sampling for over sampling rate = 4 (Case 4)

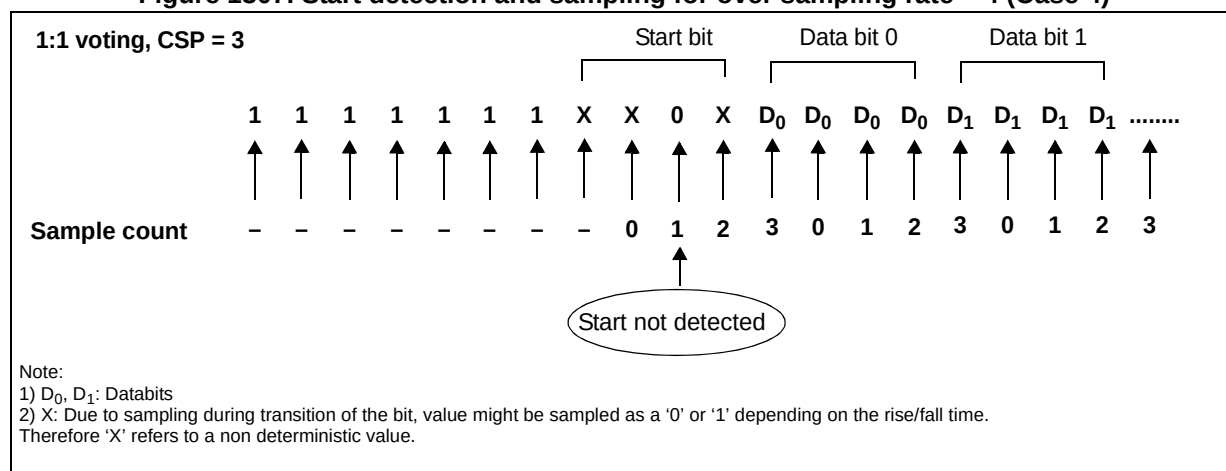
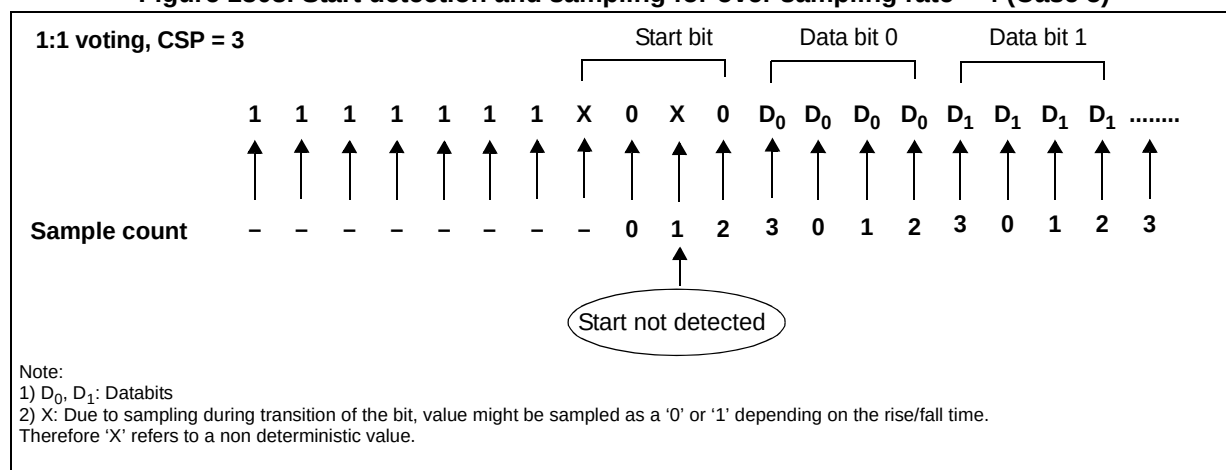


Figure 1308. Start detection and sampling for over sampling rate = 4 (Case 5)



56.3.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate register and the LIN Fraction Baud Rate register. The baud rate registers can be programmed only during Initialization mode.

The baud rate is calculated with the following formula for both receiver and transmitter:

$$\text{Equation 87} \quad T_x = R_x = \text{LIN_CLK} / (16 \times \text{LDIV})$$

Where LIN_CLK is the frequency of the baud clock (ipg_baud_clk).

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBR.

When reduced oversampling is enabled LINFBR is not used and LDIV contains only the integer part of LINIBRR.

Example 18

When ROSE = 0 (For LIN and UART mode):

LDIV = 1041.67, LIN_CLK = 80 MHz

LINIBRR = 1041d = 411 h

LINFBR = 11d = B h

Baud rate = 80 MHz / (16 × 1041.67) = 4.8 Kbit/s

Example 19

When ROSE = 1 (Only for UART mode):

LDIV = 10 d, LIN_CLK = 80 MHz

LINIBRR = 10 d, OSR = 4

Baud rate = LIN_CLK / (OSR × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s

Note:

Due to design constraints the maximum baud rate achievable is 25 Mbit/s.

The relationship "LIN_CLK > PBRIDGEx_CLK ≥ 1/3 × LIN_CLK" should also be maintained.

56.3.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescaler value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When automatic synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

56.3.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 μ s to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 μ s) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

In LINFlexD, wakeup can be detected in two ways.

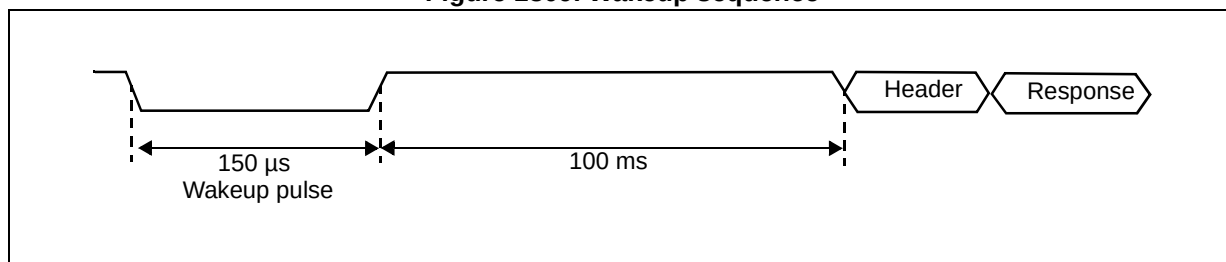
1. AUTOWU = 1

On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated (if WUPIE is set). LINFlexD is now in normal mode and ready to receive frames.

2. AUTOWU = 0

On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE is set). It is then up to the software to clear the SLEEP bit.

Figure 1309. Wakeup sequence



56.3.3 Timer

There is an 8-bit counter that is free-running (even in Sleep and Initialization modes) if it is configured in Output Compare mode. There are two software-configurable output compare registers that, upon match with the counter value, generate an output compare interrupt provided the TOCE bit in LINTCSR is set.

If this timer is configured in LIN mode, then software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state, then this counter

remains in Reset state. LIN mode has no meaning if UART is enabled — hence the counter will remain in Reset state.

56.3.4 UART mode

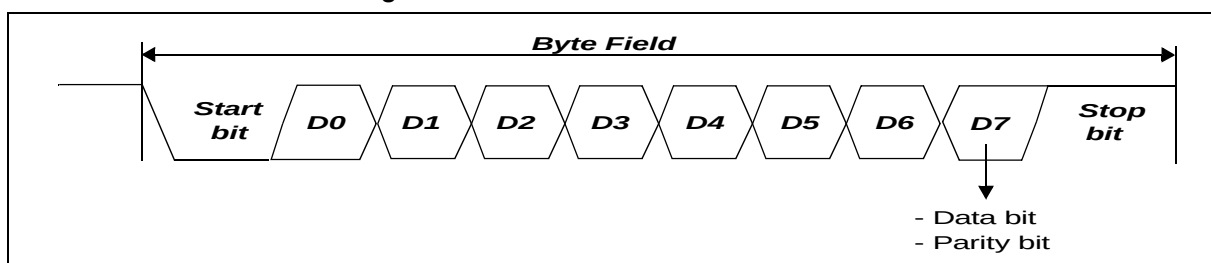
The main features in UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames, 16-bit frames, 17-bit frames
- Even/Odd/0/1 Parity
- User programmable over sampling rate to obtain the baud rate of up to 25 Mbit/s

56.3.4.1 8-bit data frames

The eighth bit can be a data bit or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

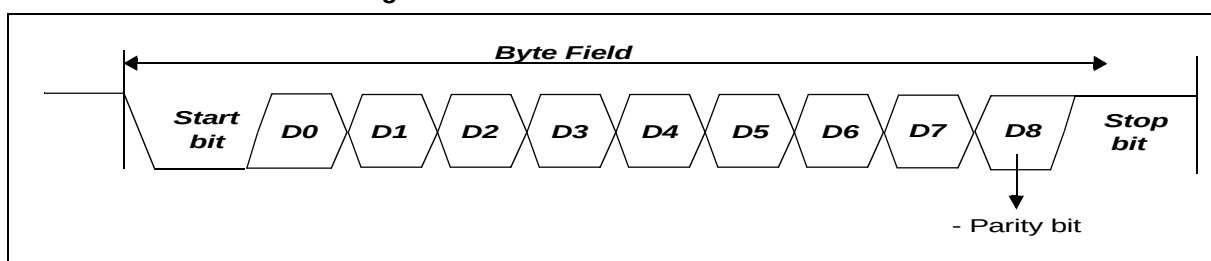
Figure 1310. UART mode 8-bit data frame



56.3.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

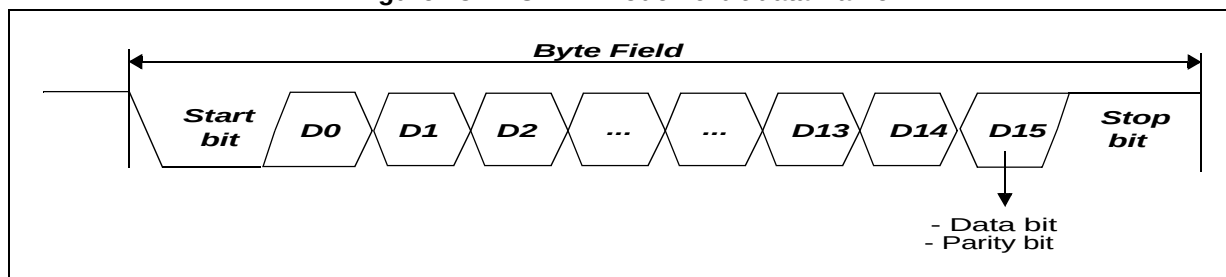
Figure 1311. UART mode 9-bit data frame



56.3.4.3 16-bit data frames

The sixteenth bit can be a data or parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

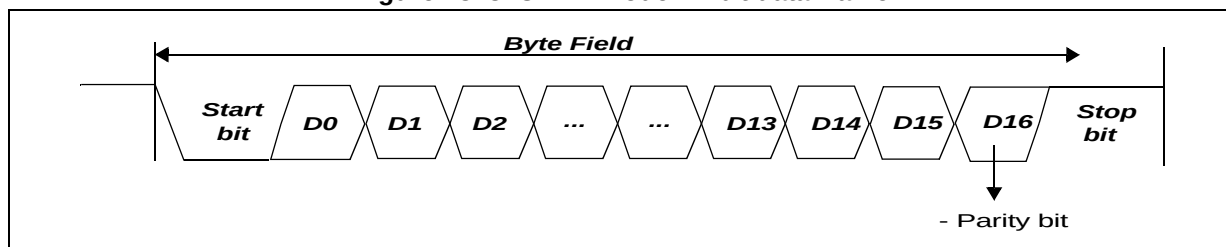
Figure 1312. UART mode 16-bit data frame



56.3.4.4 17-bit frames

The seventeenth bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

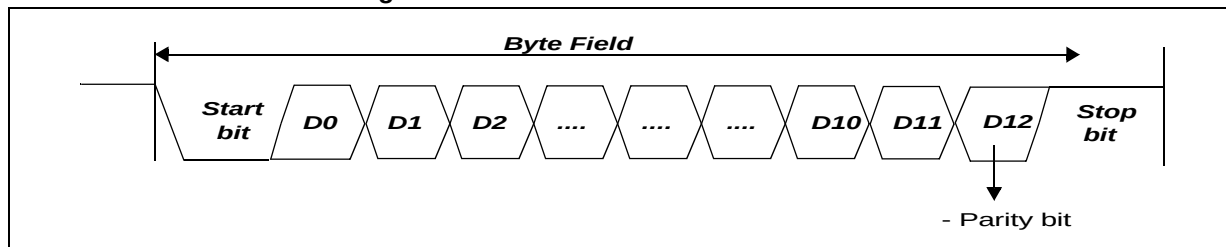
Figure 1313. UART mode 17-bit data frame



56.3.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception.

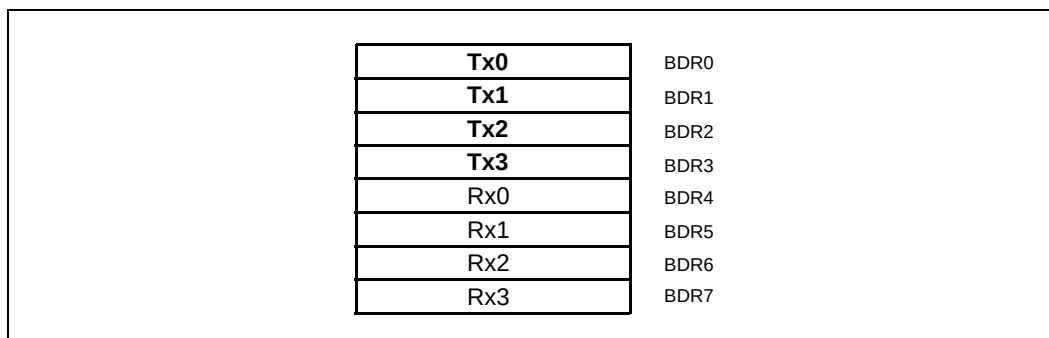
Figure 1314. UART mode 13-bit data frame



56.3.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in [Figure 1315](#).

Figure 1315. Structure of 8-byte buffer



For 16-bit frames, the lower eight bits are written in BDR0 and the upper eight bits in BDR1. The same applies for reception.

56.3.4.7 UART transmitter

In order to start transmission in UART mode, the UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the Txen bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

Note: If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Table 1249. BDRL access in UART mode

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR:WL)	IPS operation result
Write byte0	BDRL	FIFO	Byte	OK
Write byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write halfword0-1	BDRL	FIFO	Byte	IPS transfer error
Write word	BDRL	FIFO	Byte	IPS transfer error
Write byte0-1-2-3	BDRL	FIFO	Halfword	IPS transfer error
Write halfword0	BDRL	FIFO	Halfword	OK
Write halfword1	BDRL	FIFO	Halfword	IPS transfer error
Write word	BDRL	FIFO	Halfword	IPS transfer error
Read byte0-1-2-3	BDRL	FIFO	Byte/halfword	IPS transfer error
Read halfword0-1	BDRL	FIFO	Byte/halfword	IPS transfer error

Table 1249. BDRL access in UART mode (continued)

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR:WL)	IPS operation result
Read word	BDRL	FIFO	Byte/halfword	IPS transfer error
Write byte0-1-2-3	BDRL	Buffer	Byte/halfword	OK
Write halfword0-1	BDRL	Buffer	Byte/halfword	OK
Write word	BDRL	Buffer	Byte/halfword	OK
Read byte0-1-2-3	BDRL	Buffer	Byte/halfword	OK
Read halfword0-1	BDRL	Buffer	Byte/halfword	OK
Read word	BDRL	Buffer	Byte/halfword	OK

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

56.3.4.8 UART receiver

The reception of a data byte is started as soon as the user exits Initialization mode, sets the Rxen bit, and detects the start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-halfword (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. Rxen bit needs to be set only to start the reception. The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 1250. BDRM access in UART mode⁽¹⁾

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read byte4	BDRM	FIFO	Byte	OK
Read byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read halfword2-3	BDRM	FIFO	Byte	IPS transfer error
Read word	BDRM	FIFO	Byte	IPS transfer error
Read byte4-5-6-7	BDRM	FIFO	Halfword	IPS transfer error
Read halfword2	BDRM	FIFO	Halfword	OK
Read halfword3	BDRM	FIFO	Halfword	IPS transfer error
Read word	BDRM	FIFO	Halfword	IPS transfer error
Write byte4-5-6-7	BDRM	FIFO	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	FIFO	Byte/halfword	IPS transfer error
Write word	BDRM	FIFO	Byte/halfword	IPS transfer error
Read byte4-5-6-7	BDRM	Buffer	Byte/halfword	OK

Table 1250. BDRM access in UART mode⁽¹⁾ (continued)

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read halfword2-3	BDRM	Buffer	Byte/halfword	OK
Read word	BDRM	Buffer	Byte/halfword	OK
Write byte4-5-6-7	BDRM	Buffer	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	Buffer	Byte/halfword	IPS transfer error
Write word	BDRM	Buffer	Byte/halfword	IPS transfer error

1. Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and the data bits of the BDRL and BDRM registers.

Note:

- If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.
- If RDFL is programmed for a certain value but that number bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to the Idle state.
- If a STOP request arrives in the middle of a reception, it is only acknowledged after all the programmed number of data bytes have been received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout; when the state machine moves to Idle state, then only a stop request is served.
- If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if the FEIE bit in LINIER is set. As there is only one register bit for a framing error, this interrupt will be helpful in identifying which byte has the framing error.
- If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is lost depending on the RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.
- When WLS bit = 1, BDRM access depends on WL bit setting.
- When IPG_STOP is requested, for UART in FIFO mode, SW has to set the INIT bit in order to send the receiver to idle. This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

56.3.5 DMA interface

56.3.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channel where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channel where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

This release is backwards-compatible with the existing LINFlexD release if the new features are not enabled.

56.3.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in [Table 1251](#) and [Table 1252](#).

Table 1251. LINFlexD control/status fields description

Register	Field	Level	Description
LINC2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)
BIDR	DFL[5:0]	—	Data field length: – 1–8 bytes: normal frames – 1–64 bytes: extended frames
	DIR	—	Direction of the data field: 0 Reception 1 Transmission
	CCS	—	Checksum type: 0 Enhanced 1 Classic
	ID[5:0]	—	Identifier

Table 1251. LINFlexD control/status fields description (continued)

Register	Field	Level	Description
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA 0 Buffer data is free 1 Buffer data is ready
	DBFF	High	Data buffer full (8 bytes received) and DFL > 7 Used for extended frames.
	DBEF	High	Data buffer empty (8 bytes have been transmitted) and DFL > 7 Used for extended frames.
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode)
UARTCR	FBM	—	FIFO/Buffer mode 0 Buffer mode 1 FIFO mode (mandatory in DMA mode)
UARTSR	RFE	—	Rx FIFO empty 0 Rx FIFO not empty 1 Rx FIFO empty
	TFF	—	Tx FIFO full 0 Tx FIFO not full 1 Tx FIFO full
DMATXE[2**TX_CH_NUM] (1)		High	DMA Tx channel enable (read/write) In UART or LIN master mode only channel 0 can be programmed. In LIN slave mode: # DMA TX channel = IFMI[3:0] – 1. DMATXE[x] = 0b → TX channel x disabled DMATXE[x] = 1b → TX channel x enabled
DMARXE[2**RX_CH_NUM] (1)		High	DMA Rx channel enable (read/write) In UART or LIN master mode only the channel 0 can be programmed. In LIN slave mode: # DMA RX channel = IFMI[3:0] – 1. DMARXE[x] = 0b → RX channel x disabled DMARXE[x] = 1b → RX channel x enabled

1. ** stands for exponentiation

Table 1252. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0] & dma_enb_req_tx[0]	High	Logical AND between the two DMA enable bits (LINFlexD and eDMA)
DMA_TEN	LIN slave Tx	DMATXE[x] & dma_enb_req_tx[x] x = IFMI – 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0] & dma_enb_req_rx[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] & dma_enb_req_rx[x] x = IFMI – 1	High	

Control/status fields of the TCD descriptors referred to in this document are described in [Table 1253](#).

Table 1253. TCD control fields description

TCD field	Level	Description
CITER[14:0]	—	Current major iteration count
BITER[14:0]	—	Beginning major iteration count
NBYTES[31:0]	—	Inner minor byte transfer count Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset Applied to the current source address as each source read is completed.
SSIZE[2:0]	—	Source data transfer size 000 → 8-bit 001 → 16-bit 010 → 32-bit 011 → 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address
DOFF[15:0]	—	Destination address signed offset Applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 → 8-bit 001 → 16-bit 010 → 32-bit 011 → 64-bit

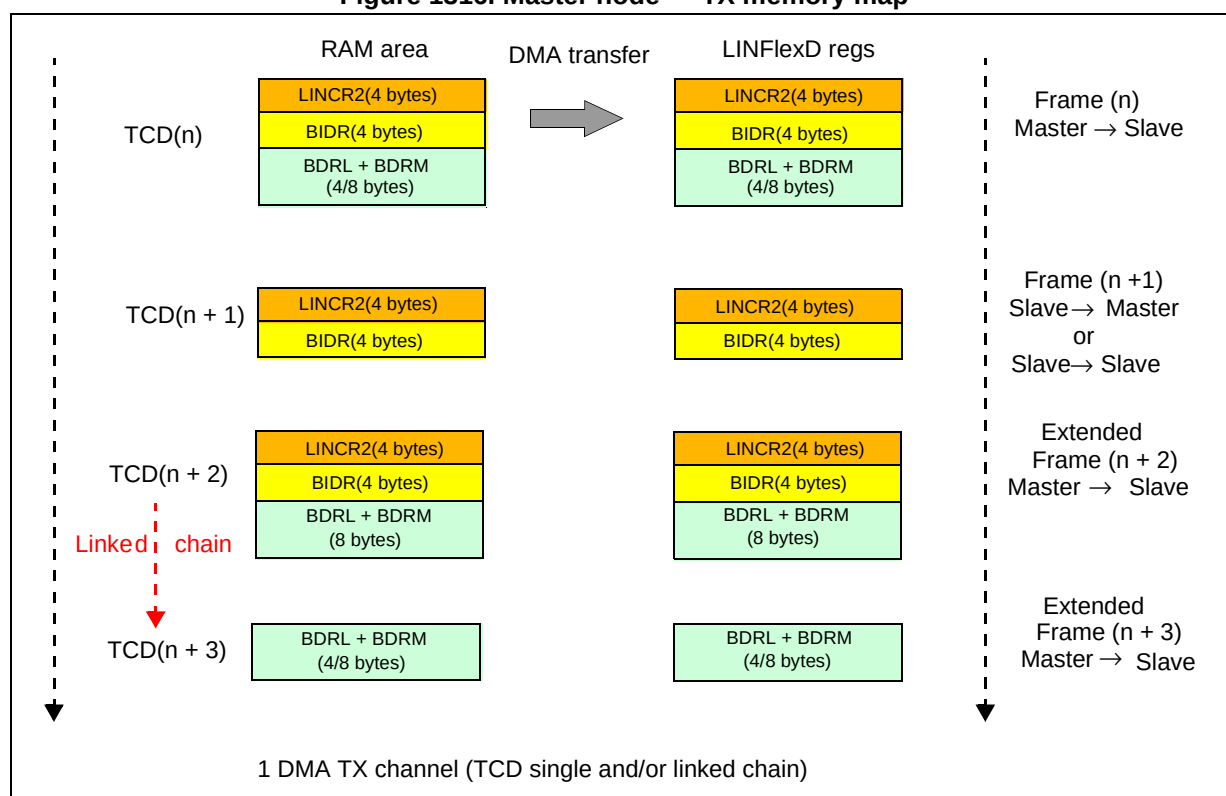
Table 1253. TCD control fields description (continued)

TCD field	Level	Description
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request
DONE	High	Channel done (the DMA has completed the outer major loop)
D_REQ	High	Disable request If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

56.3.5.3 Master node — TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in [Figure 1316](#).

Figure 1316. Master node — TX memory map



The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

[Table 1254](#) provides the register settings of the LINC2 and BIDR for each class of LIN frame.

Table 1254. Master node — Tx mode — Register setting

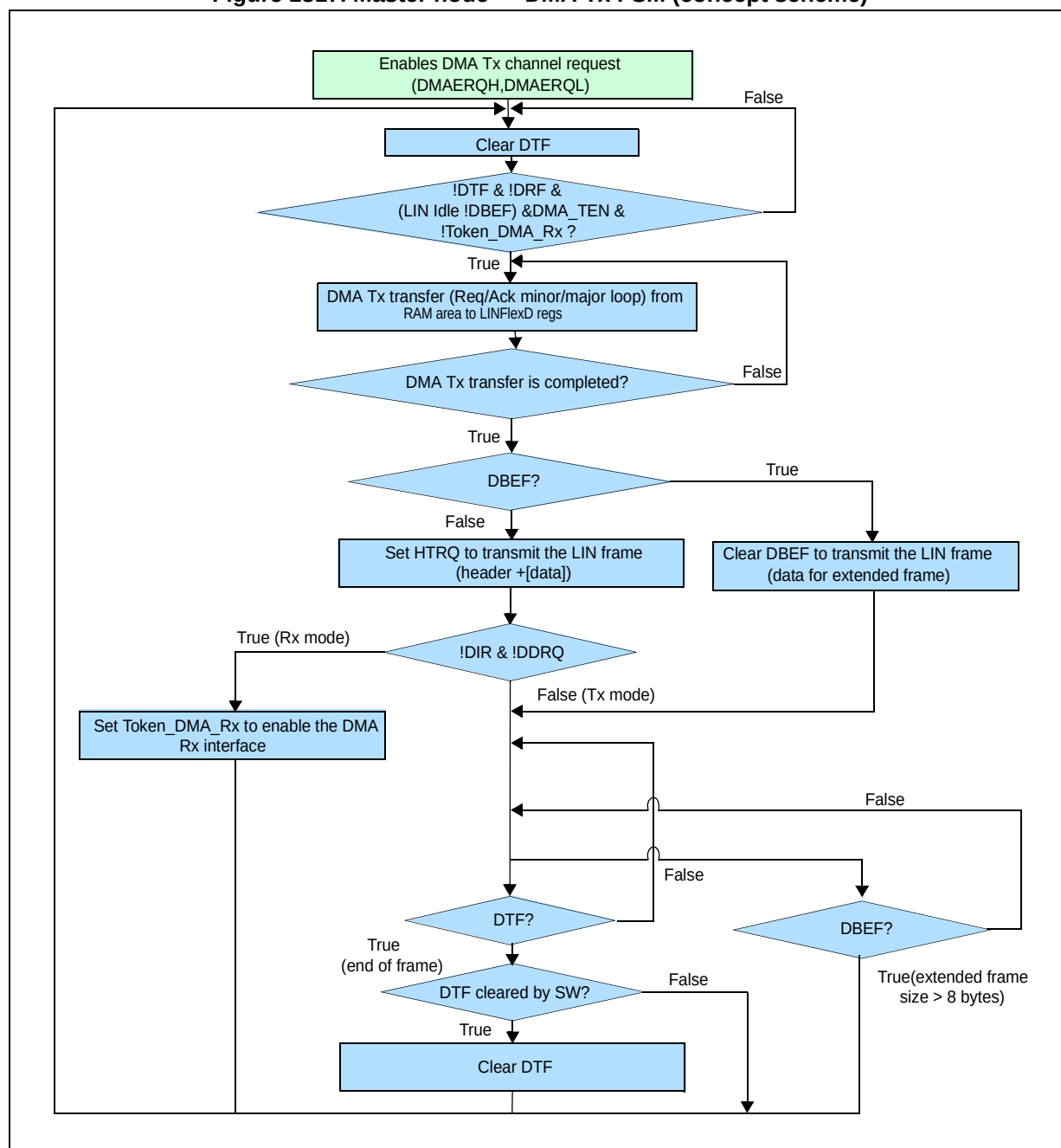
LIN frame	LINC2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in [Figure 1317](#). DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in [Table 1255](#). All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 1255. TCD setting — Master node — Tx mode

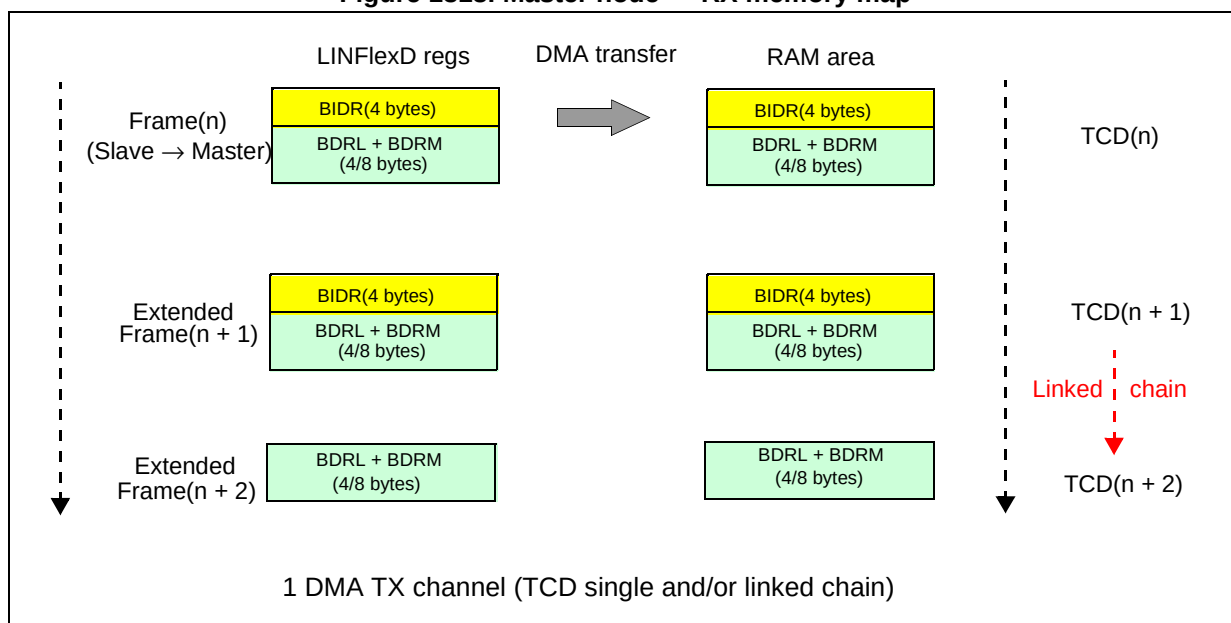
TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	$[4 + 4] + 0/4/8 = N$	Data buffer is filled with dummy bytes if length is not word-aligned LINC2 + BIDR + BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	LINC2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

Figure 1317. Master node — DMA Tx FSM (concept scheme)



56.3.5.4 Master node — RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in [Figure 1318](#).

Figure 1318. Master node — RX memory map

The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header.

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in [Figure 1319](#). DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in [Table 1256](#). All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

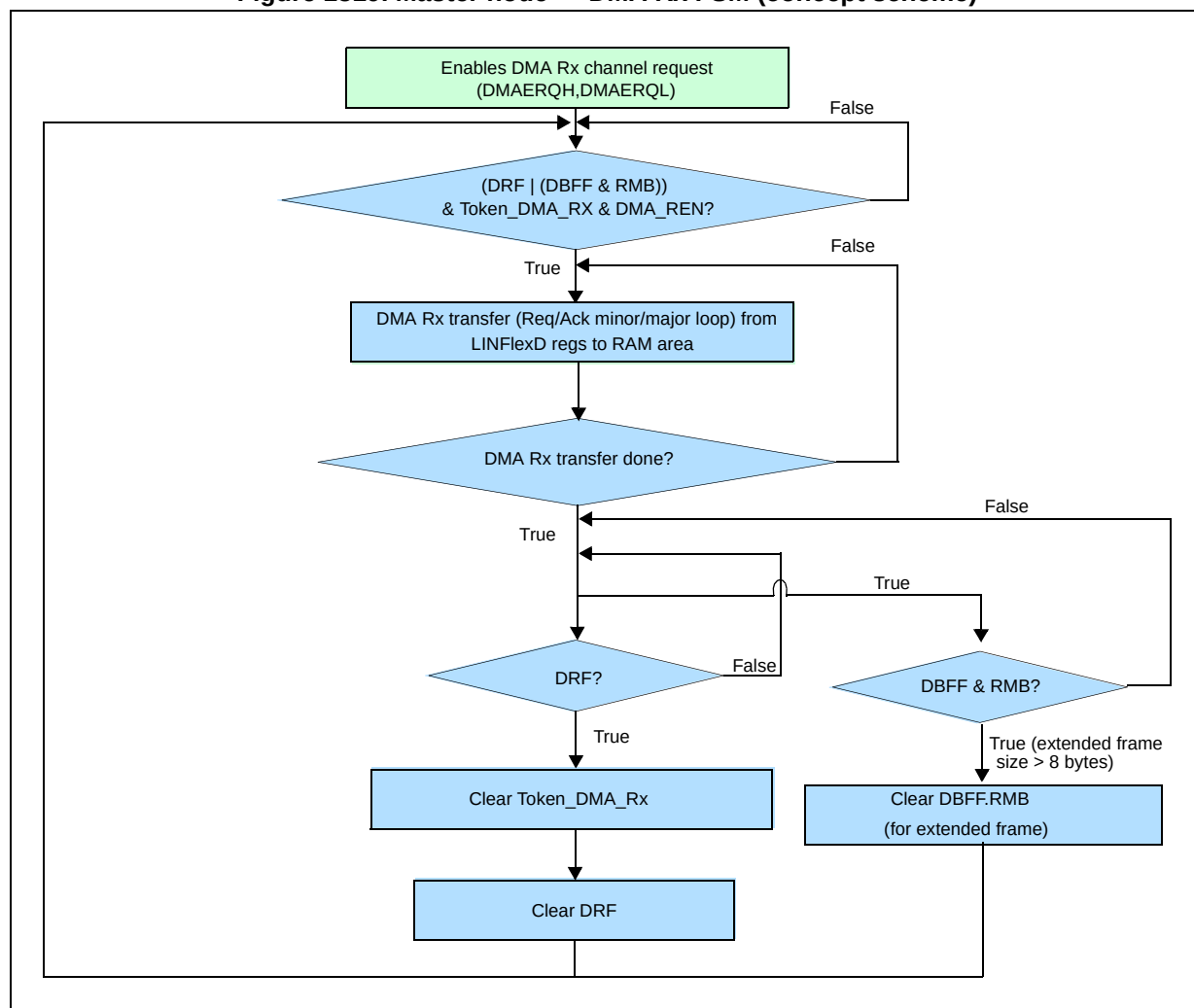
Table 1256. TCD setting — Master node — Rx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing

Table 1256. TCD setting — Master node — Rx mode (continued)

TCD field	Value	Description
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

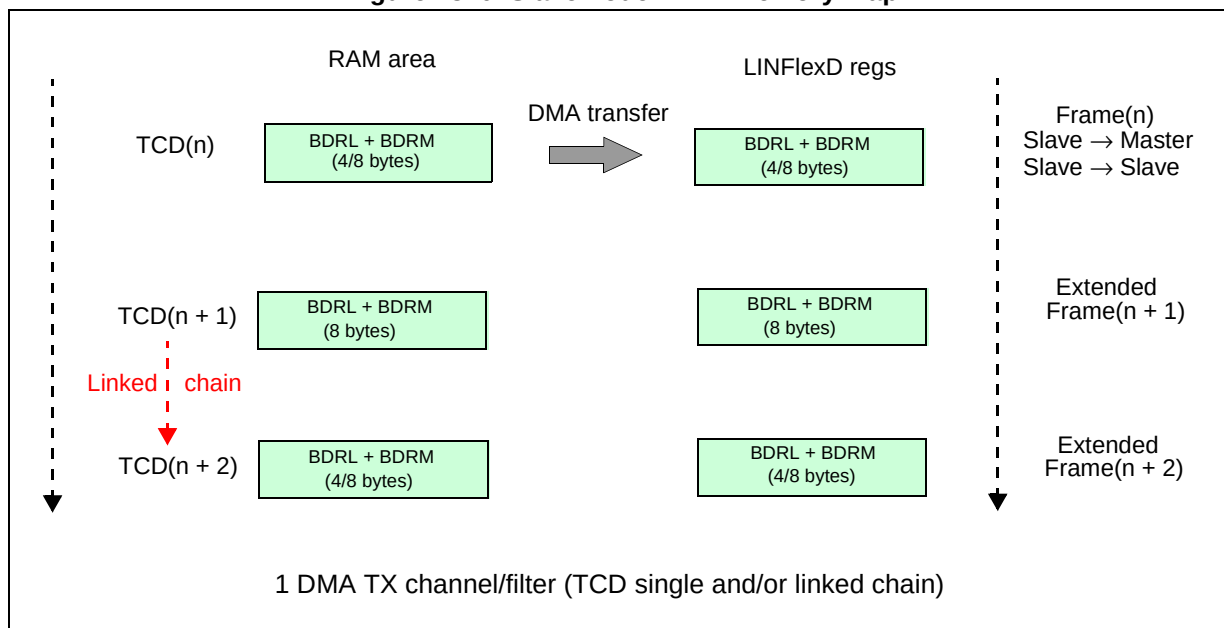
Figure 1319. Master node — DMA Rx FSM (concept scheme)



56.3.5.5 Slave node — TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in [Figure 1320](#).

Figure 1320. Slave node — TX memory map



The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in [Table 1257](#).

Table 1257. Slave node — Tx mode — Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Slave to Master or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Tx mode) for each DMA TX channel	– Identifier list mode – Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in [Figure 1321](#). DMA TX FSM moves to Idle state if $DMATXE[x] = 0$ where $x = IFMI - 1$. The TCD setting (word transfer) is given in [Table 1258](#). All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

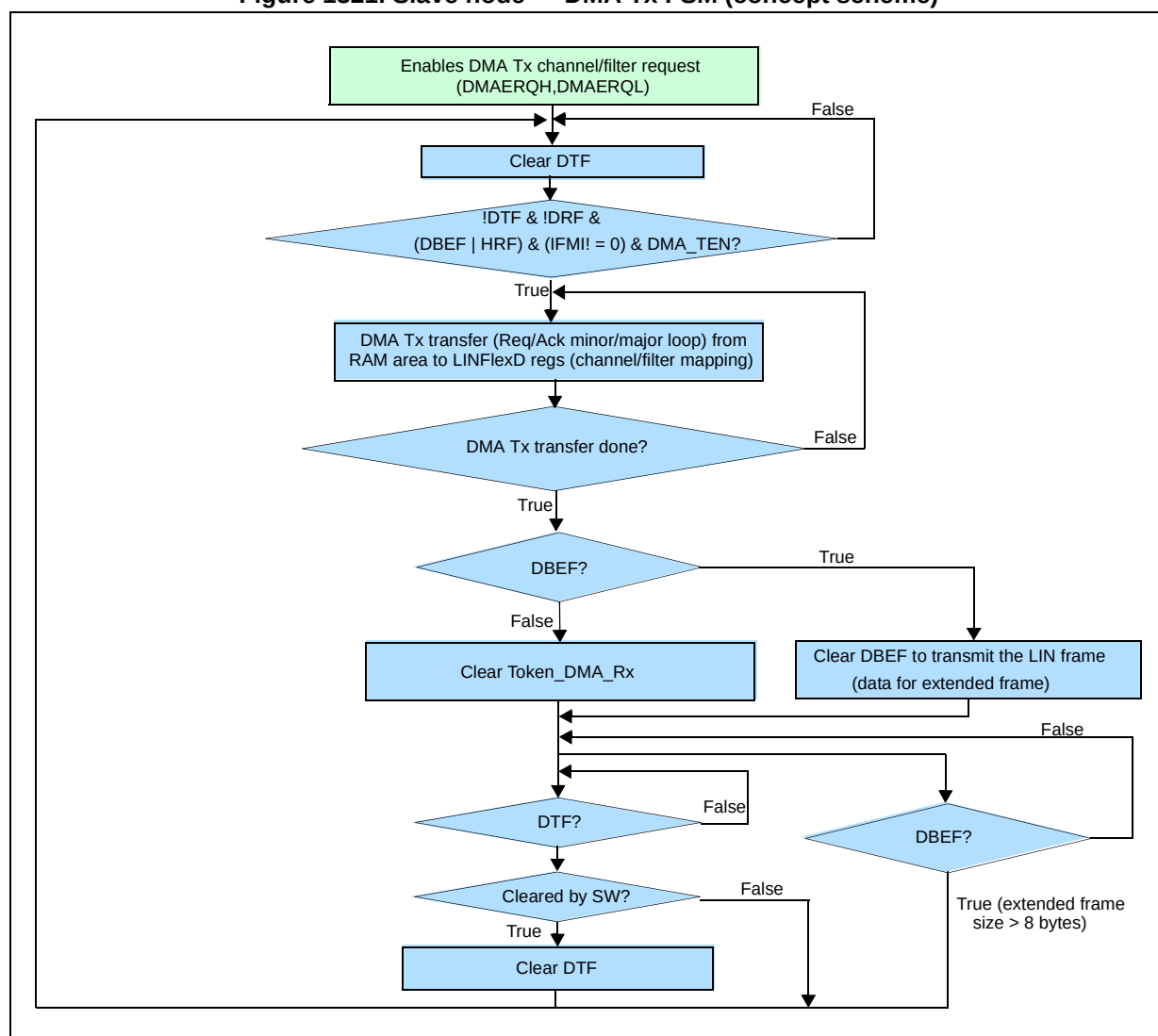
Table 1258. TCD setting — Slave node — Tx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment

Table 1258. TCD setting — Slave node — Tx mode (continued)

TCD field	Value	Description
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

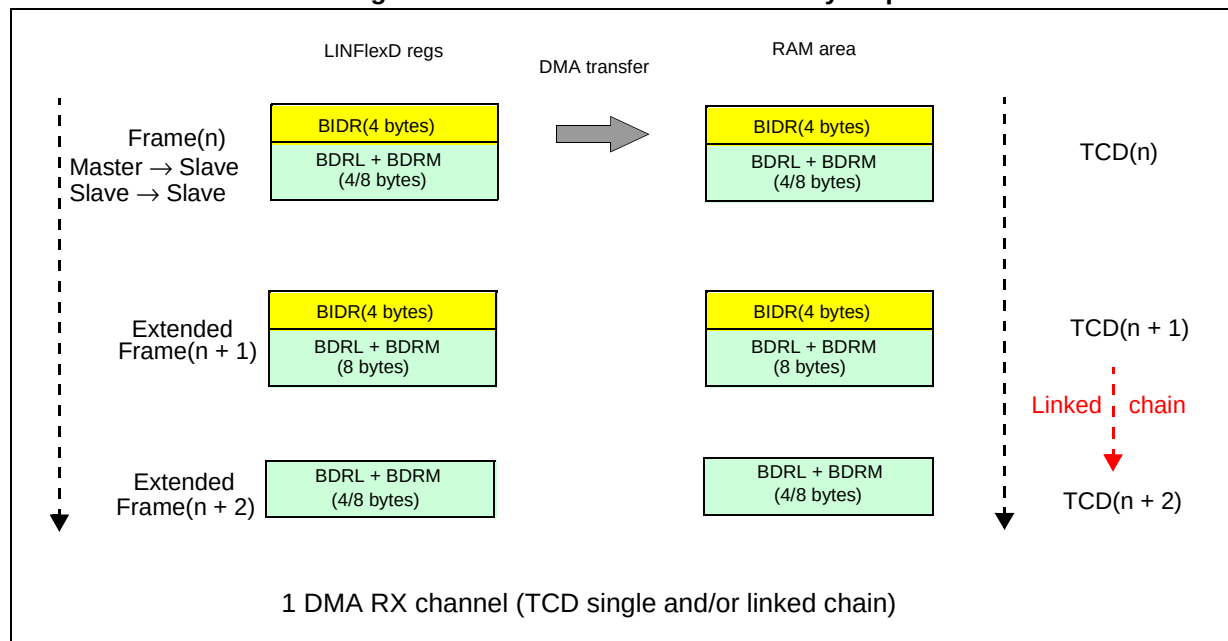
Figure 1321. Slave node — DMA Tx FSM (concept scheme)



56.3.5.6 Slave node — RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in [Figure 1322](#).

Figure 1322. Slave node — RX memory map



The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in [Table 1259](#).

Table 1259. Slave node – Rx mode — Register setting

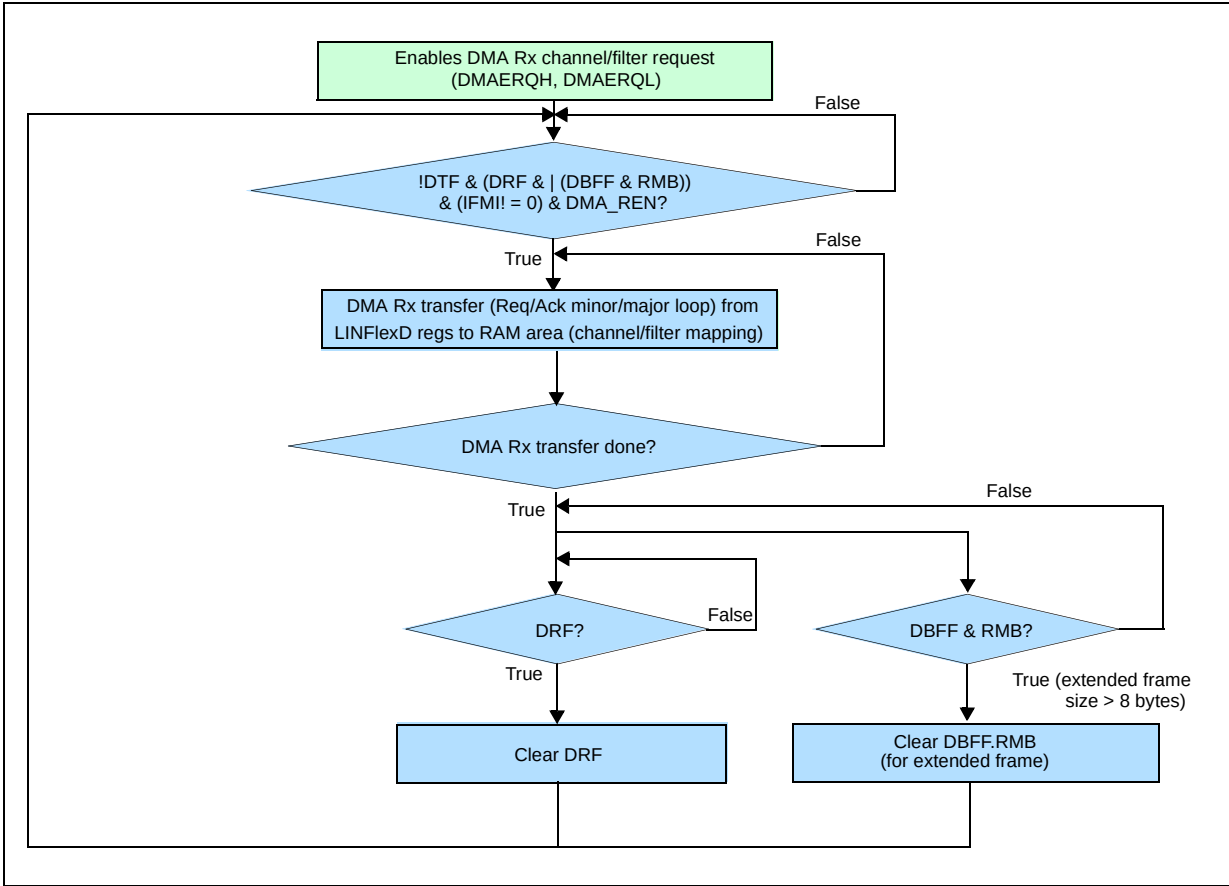
LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	– Identifier list mode – Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in [Figure 1323](#). DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in [Table 1260](#). All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 1260. TCD setting — Slave node — Rx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	$[4] + 4/8 = N$	Data buffer is filled with dummy bytes if length is not word-aligned BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

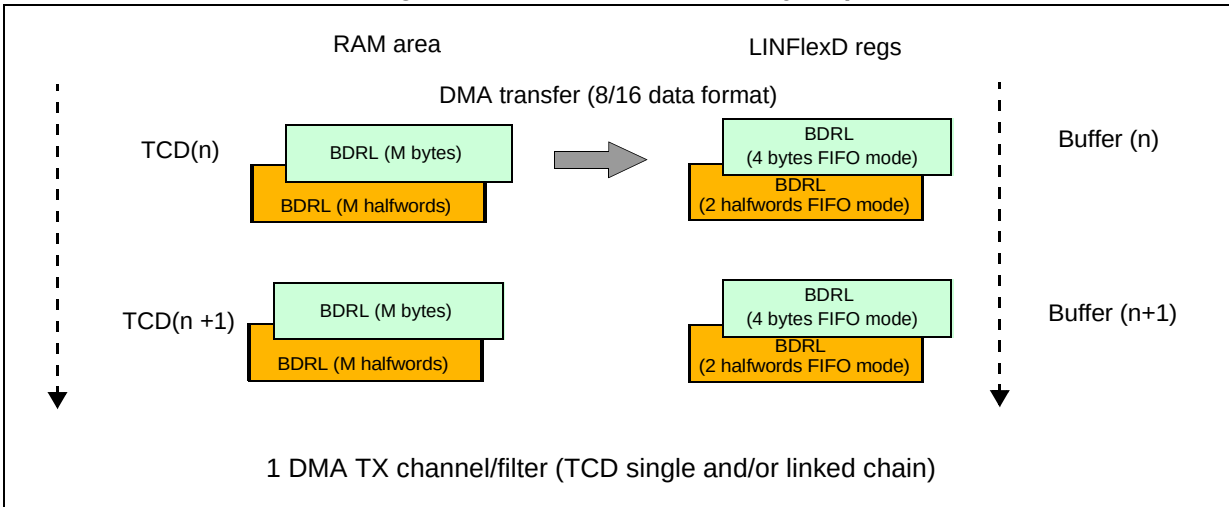
Figure 1323. Slave node — DMA Rx FSM (concept scheme)



56.3.5.7 UART — TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in [Figure 1324](#).

Figure 1324. UART — TX memory map



The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8-bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

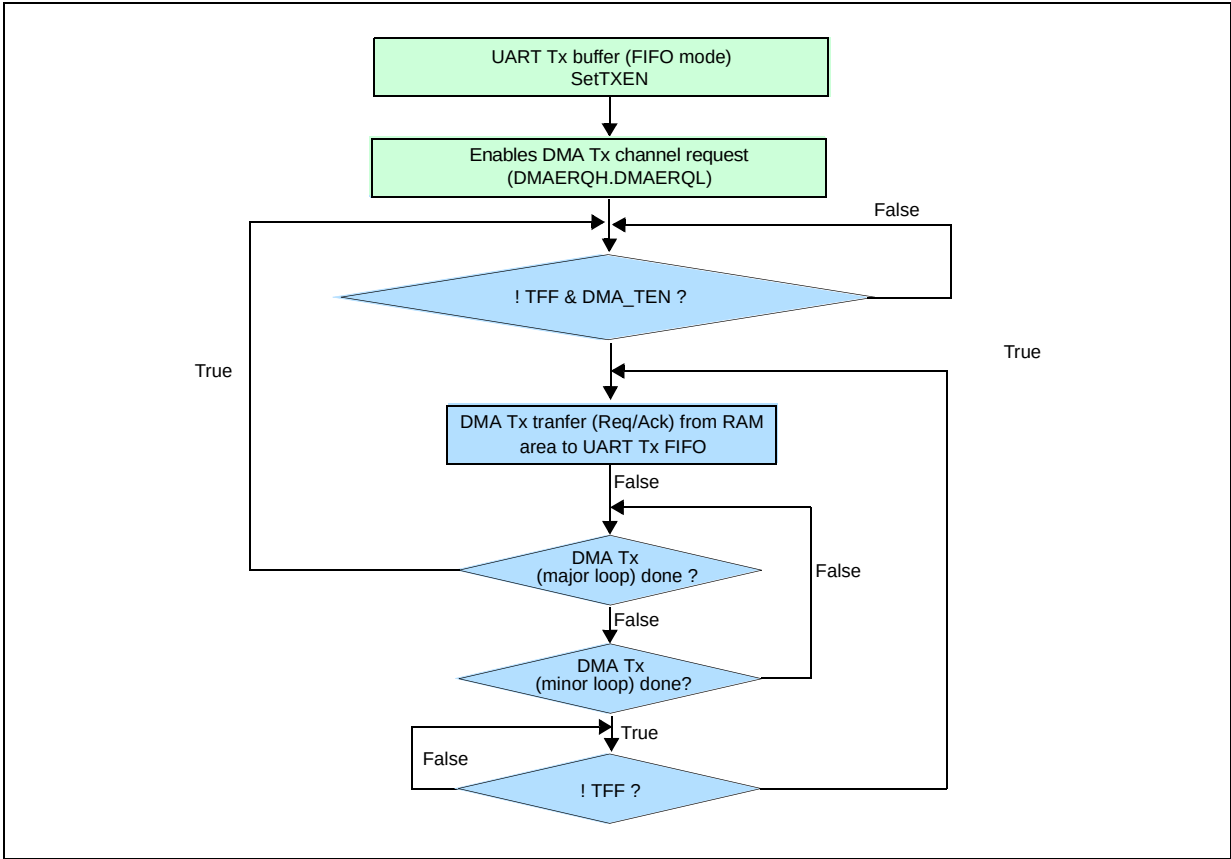
The concept FSM to control the DMA Tx interface is given in [Figure 1325](#). DMA Tx FSM will move to Idle state if DMATXE[0] = 0. The TCD setting (typical case) is given in [Table 1261](#). All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 1261. TCD setting — UART — Tx mode

TCD field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the major loop
BITER[14:0]	M		Multiple iterations for the major loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	—		RAM address
SOFF[15:0]	1	2	Byte/halfword increment
SSIZE[2:0]	0	1	Byte/halfword transfer
SLAST[31:0]	–M	–M × 2	—
DADDR[31:0] ⁽¹⁾	—		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/halfword transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain
START	0		No software request

1. DADDR[31:0] = BDRL + 0x3 for byte transfer — DADDR [31:0] = BDRL + 0x2 for halfword transfer.

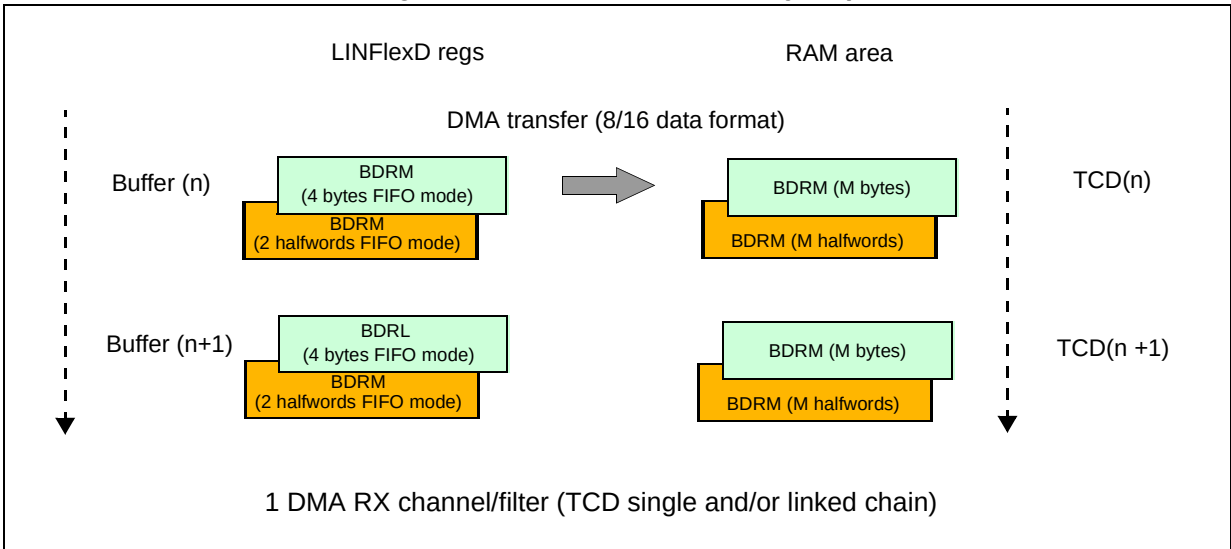
Figure 1325. UART — DMA Tx FSM (concept scheme)



56.3.5.8 UART — RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in [Table 1262](#).

Figure 1326. UART — RX memory map



The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format
- 2 halfwords in 16-bit data format

This will be sufficient because just one byte allows a reaction time of about 3.8 μ s (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

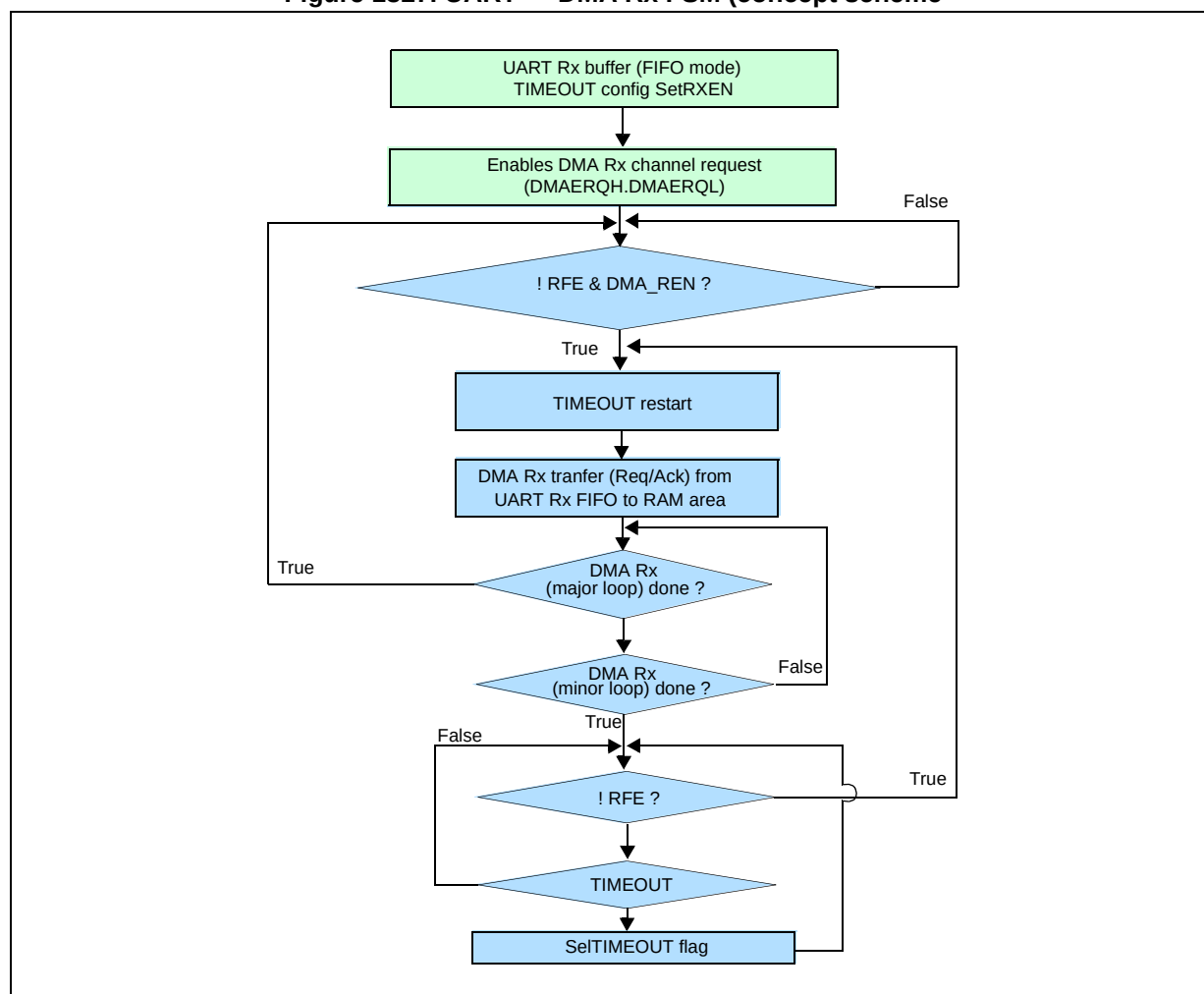
The concept FSM to control the DMA Rx interface is given in [Figure 1327](#). DMA Rx FSM will move to Idle state if DMARXE[0] = 0. The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/halfword as soon an entry is available in the Rx FIFO. A new software reset bit is required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 1262. TCD setting – UART – Rx mode

TCD field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the major loop
BITER[14:0]	M		Multiple iterations for the major loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0] ⁽¹⁾	—		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/halfword transfer
SLAST[31:0]	0		—
DADDR[31:0]	—		RAM address
DOFF[15:0]	1	2	Byte/halfword increment
DSIZE[2:0]	0	1	Byte/halfword transfer
DLAST_SGA[31:0]	–M	–M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain
START	0		No software request

1. SADDR = BDRM + 0x3 for byte transfer; SADDR[31:0] = BDRM + 0x2 for halfword transfer.

Figure 1327. UART — DMA Rx FSM (concept scheme)



56.3.5.9 DMA interface timing

The LINFlexD DMA interface includes the following signals:

- **ipd_req**
DMA peripheral request, asserted until the DMA transfer has been completed
- **dma_done_lw**
DMA minor loop done. It indicates that the channel's data transfer of nbytes (the minor loop) has been completed. It is asserted for one cycle after the data phase of the channel's last AMBA AHB write access, or when an error condition is detected
- **dma_ipd_complete**
DMA major loop finished. It is asserted for one cycle after the data phase of the last write
- **dma_ipd_ack, dma_ipd_done**
Unused in this implementation

All signals of the DMA interface must be registered in order to meet timing requirements of the DMA engine.

Figure 1328. DMA interface (major loop — single iteration)

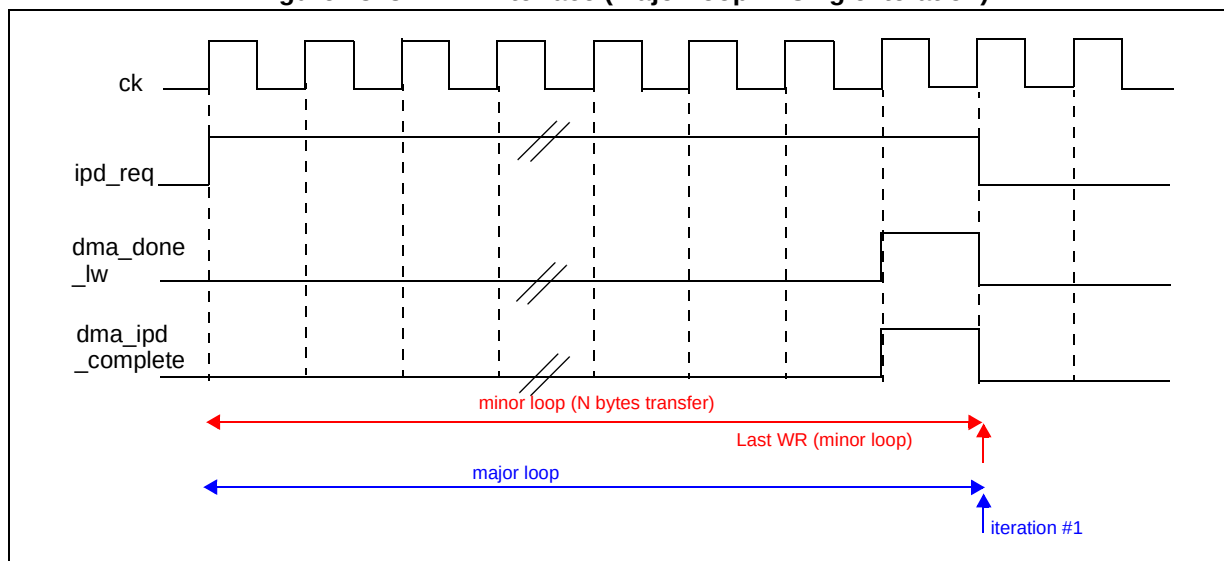
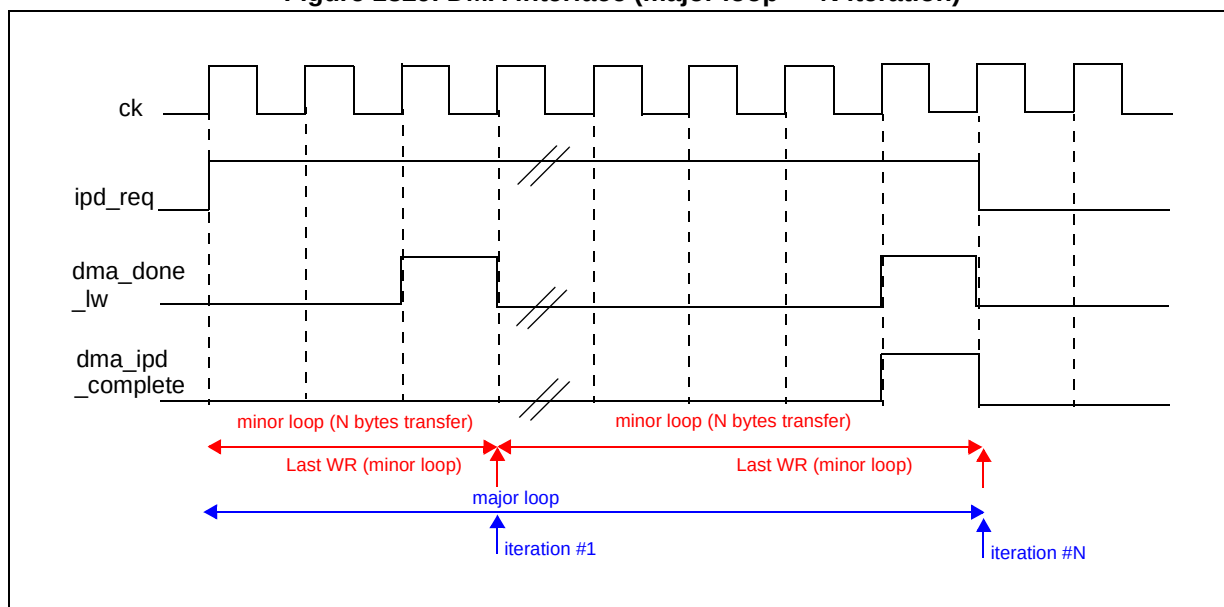


Figure 1329. DMA interface (major loop — N iteration)



56.3.5.10 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA

channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index).

- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required.
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs.
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses.
- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers.
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

56.4 Memory map and register description

56.4.1 Memory map

[Table 1263](#) shows the LINFlexD memory/register map.

Note: In Master mode, the registers IFCR0 – IFCR15 are not present.

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in ips_xfr_err being flagged.

Table 1263. LINFlexD memory map

Address offset (hex)	Register description	Register classification ⁽¹⁾	Location
00	LIN Control register 1 (LINCR1)	Saf-Relv	on page 2445
04	LIN Interrupt Enable register (LINIER)	Saf-Relv	on page 2448
08	LIN Status register (LINSR)	No-Saf	on page 2450
0C	LIN Error Status register (LINESR)	No-Saf	on page 2453
10	UART Mode Control register (UARTCR)	Saf-Relv	on page 2455
14	UART Mode Status register (UARTSR)	No-Saf	on page 2460
18	LIN Timeout Control Status register (LINTCSR)	Saf-Relv	on page 2463
1C	LIN Output Compare register (LINOOCR)	Saf-Relv	on page 2464
20	LIN Timeout Control register (LINTOCR)	Saf-Relv	on page 2464
24	LIN Fractional Baud Rate register (LINFBRR)	Saf-Relv	on page 2465

Table 1263. LINFlexD memory map (continued)

Address offset (hex)	Register description	Register classification ⁽¹⁾	Location
28	LIN Integer Baud Rate register (LINIBRR)	Saf-Relv	on page 2466
2C	LIN Checksum Field register (LINCFR)	Saf-Relv	on page 2467
30	LIN Control register 2 (LINCR2)	Saf-Relv	on page 2468
34	Buffer Identifier register (BIDR)	Saf-Relv	on page 2470
38	Buffer Data Register Least Significant (BDRL)	No-Saf	on page 2471
3C	Buffer Data Register Most Significant (BDRM)	No-Saf	on page 2472
40	Identifier Filter Enable register (IFER)	Saf-Relv	on page 2473
44	Identifier Filter Match index (IFMI)	Saf-Relv	on page 2474
48	Identifier Filter Mode register (IFMR)	Saf-Relv	on page 2474
4C	Identifier Filter Control registers (IFCR0)	Saf-Relv	on page 2476
50	Identifier Filter Control registers (IFCR1)	Saf-Relv	on page 2476
54	Identifier Filter Control registers (IFCR2)	Saf-Relv	on page 2476
58	Identifier Filter Control registers (IFCR3)	Saf-Relv	on page 2476
5C	Identifier Filter Control registers (IFCR4)	Saf-Relv	on page 2476
60	Identifier Filter Control registers (IFCR5)	Saf-Relv	on page 2476
64	Identifier Filter Control registers (IFCR6)	Saf-Relv	on page 2476
68	Identifier Filter Control registers (IFCR7)	Saf-Relv	on page 2476
6C	Identifier Filter Control registers (IFCR8)	Saf-Relv	on page 2476
70	Identifier Filter Control registers (IFCR9)	Saf-Relv	on page 2476
74	Identifier Filter Control registers (IFCR10)	Saf-Relv	on page 2476
78	Identifier Filter Control registers (IFCR11)	Saf-Relv	on page 2476
7C	Identifier Filter Control registers (IFCR12)	Saf-Relv	on page 2476
80	Identifier Filter Control registers (IFCR13)	Saf-Relv	on page 2476
84	Identifier Filter Control registers (IFCR14)	Saf-Relv	on page 2476
88	Identifier Filter Control registers (IFCR15)	Saf-Relv	on page 2476
8C ⁽²⁾	Global Control register (GCR)	Saf-Relv	on page 2478
90 ⁽²⁾	UART Preset Timeout register (UARTPTO)	Saf-Relv	on page 2480
94 ⁽²⁾	UART Current Timeout register (UARTCTO)	No-Saf	on page 2480
98 ⁽²⁾	DMA Tx Enable register (DMATXE)	Saf-Relv	on page 2482
9C ⁽²⁾	DMA Rx Enable register (DMARXE)	Saf-Relv	on page 2483

1. No-Saf = Not safety relevant, Saf-Lat = Safety Latent, Saf-Relv = Safety Relevant, Saf-Crit = Safety Critical.

2. The address offset depends on the no_of_filters and value of slave. Refer to the device configuration chapter for the number of filters and value of slave used in this device.

56.4.2 Register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of generic slave = 0 and slave = 1.

56.4.2.1 LIN Control Register 1 (LINC1)

LINC1 consists of control bits used to configure features of the LINFlexD.

Figure 1330. LIN Control Register 1 (LINC1)

Address: 0000h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD ⁽¹⁾	CFD ⁽¹⁾	LASE ⁽¹⁾	AUTOWU ⁽¹⁾	MBL3 ⁽¹⁾	MBL2 ⁽¹⁾	MBL1 ⁽¹⁾	MBL0 ⁽¹⁾	BF ⁽¹⁾	0	LBKM ⁽¹⁾	MME ⁽¹⁾	SSBL ⁽¹⁾	RBLM ⁽¹⁾	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0/1 ⁽²⁾	0	0	1	0

1. Register bit can be read in any mode, written only in Initialization mode.

2. When slave = 0, then this bit will read a 1 always and cannot be programmed.

Table 1264. LINC1 field descriptions

Field	Description
0–15	Reserved Read returns 0
16 CCD	Checksum Calculation disable 0 Checksum calculation is done by hardware When this bit is reset the LINC1FR register is read-only. 1 Checksum calculation is disabled When this bit is set the LINC1FR register is read/write. User can program this register to send a software calculated checksum/CRC (provided CFD is reset). This bit can be written during Initialization mode only. It is read-only in Normal mode.
17 CFD	Checksum field disable 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame This bit can be configured during Initialization mode only. It is read-only in Normal mode.

Table 1264. LINCR1 field descriptions (continued)

Field	Description																																		
18 LASE	LIN Auto Synchronization Enable 0 Auto synchronization disabled 1 Auto synchronization enabled This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0).																																		
19 AUTOWU	Auto Wakeup 0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set This bit can be configured during Initialization mode only. This bit is utilized in UART mode also.																																		
20–23 MBL[3:0]	Master Break Length These bits choose the length of the Sync break to be generated by the master. <table border="1"> <thead> <tr> <th>MBL[3:0]</th><th>Break length</th></tr> </thead> <tbody> <tr><td>0</td><td>10-bit</td></tr> <tr><td>1</td><td>11-bit</td></tr> <tr><td>2</td><td>12-bit</td></tr> <tr><td>3</td><td>13-bit</td></tr> <tr><td>4</td><td>14-bit</td></tr> <tr><td>5</td><td>15-bit</td></tr> <tr><td>6</td><td>16-bit</td></tr> <tr><td>7</td><td>17-bit</td></tr> <tr><td>8</td><td>18-bit</td></tr> <tr><td>9</td><td>19-bit</td></tr> <tr><td>A</td><td>20-bit</td></tr> <tr><td>B</td><td>21-bit</td></tr> <tr><td>C</td><td>22-bit</td></tr> <tr><td>D</td><td>23-bit</td></tr> <tr><td>E</td><td>36-bit (cooling)</td></tr> <tr><td>F</td><td>50-bit</td></tr> </tbody> </table> These bits can be programmed in Initialization mode only. They are read-only in Normal mode.	MBL[3:0]	Break length	0	10-bit	1	11-bit	2	12-bit	3	13-bit	4	14-bit	5	15-bit	6	16-bit	7	17-bit	8	18-bit	9	19-bit	A	20-bit	B	21-bit	C	22-bit	D	23-bit	E	36-bit (cooling)	F	50-bit
MBL[3:0]	Break length																																		
0	10-bit																																		
1	11-bit																																		
2	12-bit																																		
3	13-bit																																		
4	14-bit																																		
5	15-bit																																		
6	16-bit																																		
7	17-bit																																		
8	18-bit																																		
9	19-bit																																		
A	20-bit																																		
B	21-bit																																		
C	22-bit																																		
D	23-bit																																		
E	36-bit (cooling)																																		
F	50-bit																																		
24 BF	By-pass filter 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter This bit can be programmed during Initialization mode only. <i>Note: If generic_slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed.</i>																																		
25	Reserved																																		

Table 1264. LINCR1 field descriptions (continued)

Field	Description
26 LBKM	Loop Back mode 0 Loop Back Mode disabled 1 Loop Back mode enabled Refer to Section 56.3.2.2, Test mode. Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also.
27 MME	Master mode enable 0 Slave Mode 1 Master Mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. <i>Note: If generic slave = 0, then this bit will read a '1' always, and cannot be programmed.</i>
28 SSBL	Slave Mode Sync Break Length 0 11 bit break length 1 10 bit break length This bit can be programmed in Initialization mode only. It is read-only in Normal mode.
29 RBLM	Receiver Buffer Locked mode 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software. This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also. When in_ps_bit_remap is set, SLEEP bit can be set/cleared by in_ps_sleep signal only.
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also. When in_ps_bit_remap is set, INIT bit can be set/cleared by in_ps_init signal only.

56.4.2.2 LIN Interrupt enable register (LINIER)

Figure 1331. LIN Interrupt enable register (LINIER)

Address: 0004h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R						0	0									
W	SZIE	OCIE	BEIE	CEIE	HEIE			FEIE	BOIE	LSIE	WUIE	DBFIE	DBIE/TOIE	DRIE	DTIE	HRIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1265. LINIER field descriptions

Field	Description
0–15	Reserved Read returns 0
16 SZIE	Stuck at Zero Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set.
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR.
20 HEIE	Header Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. <i>Note: If generic slave = 0, then this bit will always read a 0 and cannot be programmed.</i>
21–22	Reserved Read returns 0.

Table 1265. LINIER field descriptions (continued)

Field	Description
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR.
25 LSIE	LIN state Interrupt enable 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag.
26 WUIE	Wakeup interrupt enable 0 No interrupt 1 Interrupt enabled If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated.
27 DBFIE	Data Buffer Full Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the DBFF bit in LINSR is also set.
28 DBEIE/TOIE	Data Buffer Empty Interrupt enable/Timeout Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and LINSR[DBEF] status bit is set (in LIN mode) or if this bit is set and UARTSR[TO] status bit is set (in UART mode).
29 DRIE	Data Reception complete Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set.
30 DTIE	Data Transmitted Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set.
31 HRIE	Header Received Interrupt 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. <i>Note: If generic slave = 0, then this bit will always read a 0, and cannot be programmed.</i>

56.4.2.3 LIN Status Register (LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Figure 1332. LIN Status Register (LINSR)

Address: 0x0008h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	AUTOSYNC_COMP	RDC		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS3	LINS2	LINS1	LINS0	0	0	RMB	DRBNE	RXbusy	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
W							W1C	W1C			W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 1266. LINSR field description

Field	Description
0–11	Reserved Read returns 0
12 AUTOSYNC_C OMP	AUTOSYNC_COMP This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers can be read in autosynchronization mode.
13–15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table 1266. LINSR field description (continued)

Field	Description
16–19 LINS[3:0]	LIN state 0: Sleep mode LINFlexD is in Sleep mode, to save power consumption. 1: Init, 2: Idle This mode is entered when: – SLEEP bit and INIT bit are reset by software. – Wakeup pulse has been received on RX pin (AUTOWU set). – Previous frame transmission/reception has been completed/aborted. 3: Sync Break In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. <i>Note: In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle.</i> 4: Sync Del In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 5: Sync Field In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 6: Identifier Field In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 7: Header Reception/Transmission In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 8: Data Reception/Data Transmission In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 9: <i>Checksum</i> Data transmission/reception completed, checksum transmission/reception ongoing. In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits.
20–21	Reserved Read returns 0.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.

Table 1266. LINSR field description (continued)

Field	Description
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.
24 RXbusy	Receiver Busy flag 0 Receiver is idle 1 Reception ongoing In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit.
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. <i>Note: After reset is released, RDI reflects the actual value of Rx pin.</i>
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin. 1. When slave is in Sleep mode 2. When master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27 DBFF	Data Buffer full flag This bit is set by hardware when the buffer is full (8 bytes received) and DFL > 7. It should be cleared by software. This bit is also reset by hardware in Initialization mode.
28 DBEF	Data Buffer empty flag This bit is set by hardware when the buffer is empty (8 bytes have been transmitted) and DFL > 7. It should be cleared by software after data buffer is refilled by software. Transmission of next 8 bytes will not start unless this bit is reset by software. This bit is also reset by hardware in Initialization mode.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. <i>Note: In case framing error or checksum error occurs then this flag is not set.</i>

Table 1266. LINSR field description (continued)

Field	Description
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. <i>Note: For LIN mode, in case a bit error occurs (and IOBE is reset) then this flag is not set.</i>
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: a) All filters are inactive and Bypass filter (BF) bit is set b) No match in any filter and Bypass filter (BF) bit is set c) TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. <i>Note: If generic slave = 0, then this bit will always read a 0, and cannot be programmed.</i>

56.4.2.4 LIN Error Status Register (LINESR)

Figure 1333. LIN error status register (LINESR)

Address: 000Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	BEF	CEF	SLEF	SDEF	IDPEF	FEF	BOF	0	0	0	0	0	0	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c							w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 56.3.2.5, Errors](#) for detailed description of all errors.

Table 1267. LINESR field descriptions

Field	Description
0–15	Reserved Read returns 0
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.

Table 1267. LINESR field descriptions (continued)

Field	Description
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of OCR1 or OCR2 This bit should be cleared by software. If this bit is set and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINC1R1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. If generic <i>slave</i> = 0, then this bit will always read a 0, and cannot be programmed.
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. <i>Note: If generic slave = 0, then this bit will always read a 0, and cannot be programmed.</i>
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. <i>Note: Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set.</i> If generic <i>slave</i> = 0, then this bit will always read a 0, and cannot be programmed. For generic <i>slave</i> = 1, this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.

Table 1267. LINESR field descriptions (continued)

Field	Description
25–30	Reserved Read returns 0.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

56.4.2.5 UART Mode Control Register (UARTCR)

Figure 1334. UART Mode Control Register (UARTCR)

Address: 0010h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIS ⁽¹⁾	CSP ⁽¹⁾			OSR ⁽¹⁾				ROSE ⁽¹⁾	NEF ⁽²⁾			DTU/PCE_Tx ⁽¹⁾	SBUR[1:0] ⁽¹⁾		WLS ⁽¹⁾
W																
Reset	0	0			0				0	0	0	0	0	0		0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	TDFL ^{(2)/TFC⁽³⁾}				RDFL ^{(2)/RFC⁽³⁾}				RFBM ⁽¹⁾	TFBM ⁽¹⁾	WL1 ⁽¹⁾	PC1 ⁽¹⁾	RxEn	TxEn	PC0 ⁽¹⁾	PCE ⁽¹⁾	WLO ⁽¹⁾	UART ⁽¹⁾
W																		
Reset	0				0				0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.

2. Register bit is a read/write field.

3. Register bit is read-only field.

Table 1268. UARTCR field descriptions

Field	Description										
0 MIS	Monitor Idle State 1UARTCTO monitors the idle state of the reception line. 0UARTCTO monitors the number of bits to be received. The default value of this bit is 0.										
1–3 CSP	Configurable Sample Point (i) These bits will decide the sample point during reduced over sampling (ROSE=1). CSP can take the following range of values for a certain over sampling rate: <table border="1"> <thead> <tr> <th>OSR</th><th>CSP</th></tr> </thead> <tbody> <tr> <td>4</td><td>2, 3</td></tr> <tr> <td>5</td><td>2, 3, 4</td></tr> <tr> <td>6</td><td>3, 4, 5</td></tr> <tr> <td>8</td><td>NA</td></tr> </tbody> </table> For any values of CSP other than the ones mentioned in the above table, data sampled is erroneous.	OSR	CSP	4	2, 3	5	2, 3, 4	6	3, 4, 5	8	NA
OSR	CSP										
4	2, 3										
5	2, 3, 4										
6	3, 4, 5										
8	NA										
4–7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled (ROSE=1). For proper functionality, OSR can only take values 4, 5, 6, and 8. For any other values start bit of data is not detected by receiver and thus the reception will never start.										
8 ROSE	Reduced Over Sampling Enable 0Each bit is over sampled sixteen times. 1OSR bits decide the over sampling rate. The default value of this bit is 0.										
9–11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received.										

Table 1268. UARTCR field descriptions (continued)

Field	Description
12 DTU/PCE_Tx	Disable Timeout in UART mode/Parity transmission and checking When in_ps_bit_remap is set: 0 Parity transmission and checking disabled 1 Parity transmission and checking enabled When in_ps_bit_remap is cleared: 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received This bit can be programmed in initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received. In Buffer mode: The timer gets reset whenever the number of bytes received (rec_byte_cnt) is equal to RDFL. At this point the DRF bit is also set and num_of_frames (number of frames received), rec_byte_cnt are reset. When num_of_frames equals NEF, Disable Timeout is generated. But this clears num_of_frames also. Thus the timer restarts again. The value of counter is 0 when it restarts. In FIFO mode: When num_of_frames equals NEF, Disable Timeout is generated which clears num_of_frames and causes the timer to restart again. The value of counter is 0 when it restarts. <i>Note:</i> – Disable Timeout causes only the Timer to restart which enables Timeout again – Timer reset means it resets and starts counting again and the Timeout is enabled.
13–14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 Reserved This bit can be programmed in the initialization mode only, when the UART bit is set.
15 WLS	Special Word Length in UART mode 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support. This bit can be programmed in initialization mode only when the UART bit is set. Parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode.

Table 1268. UARTCR field descriptions (continued)

Field	Description
16–18 TDFL/TFC	Transmitter Data Field Length/TX FIFO Counter TDFL defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL is a read/write field. Bit 15 is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations TDFL = x01 or TDFL = x11 are allowed. x00 1 byte x01 2 bytes x10 3 bytes x11 4 bytes TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TFCx is a read-only field available for debug purposes. 000 empty 001 1 byte 010 2 bytes 011 3 bytes 100 4 bytes Others: Reserved TDFL/TFC can be programmed and are significative only when the UART bit is set.
19–21 RDFL/RFC	Reception Data Field Length /RX FIFO Counter RDFL defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations RDFL = x01 or RDFL = x11 are allowed. x00 1 byte x01 2 bytes x10 3 bytes x11 4 bytes RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RFCx is a read-only field available for debug purposes. 000 empty 001 1-byte 010 2 bytes/1.5 byte in case of WLS bit setting 011 3 bytes 100 4 bytes/2×1.5 byte in case of WLS bit setting Others: reserved RDFL/RFC can be programmed and are significative only when the UART bit is set. <i>Note: In buffer mode, RDFL should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint.</i>
22 RFBM	RFBM Rx Fifo/Buffer mode 0 Rx Buffer mode enabled 1 Rx Fifo mode enabled (mandatory in DMA Rx mode) This bit can be programmed in Initialization mode, only when the UART bit is set.
23 TFBM	Tx Fifo/Buffer mode 0 Tx Buffer mode enabled 1 Tx Fifo mode enabled (mandatory in DMA Tx mode) This bit can be programmed in initialization mode, only when the UART bit is set.

Table 1268. UARTCR field descriptions (continued)

Field	Description
24 WL1	Word Length in UART mode 00 7 bits data + parity 01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1 10 15 bits data + parity 11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1 This bit can be programmed in Initialization mode only, when the UART bit is set.
25 PC1	Parity Control 00 Parity sent is Even 01 Parity sent is Odd 10 A logical 0 is always transmitted/checked as parity bit 11 A logical 1 is always transmitted/checked as parity bit This bit can be programmed in Initialization mode, only when UART bit is set.
26 RxEn	Receiver Enable 0 Receiver disabled 1 Receiver enabled This bit can be programmed only when the UART bit is set.
27 TxEn	Transmitter Enable 0 Transmitter disabled 1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed This bit can be programmed only when UART bit is set.
28 PC0	Parity Control 00 Parity sent is Even 01 Parity sent is Odd 10 A logical 0 is always transmitted/checked as parity bit 11 A logical 1 is always transmitted/checked as parity bit This bit can be programmed in Initialization mode, only when UART bit is set.
29 PCE	Parity Control Enable When in_ps_bit_remap is set: 0 Parity reception and checking disabled 1 Parity reception and checking enabled When in_ps_bit_remap is cleared: 0 Parity transmission, reception, and checking disabled 1 Parity transmission, reception, and checking enabled This bit can be programmed in Initialization mode only, when the UART bit is set.
30 WL0	Word Length in UART mode 00 7 bits data + parity 01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1 10 15 bits data + parity 11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1 This bit can be programmed in Initialization mode only, when UART bit is set.
31 UART	UART Mode 0 LIN mode 1 UART mode This bit can be programmed in Initialization mode only.

56.4.2.6 UART Mode Status Register (UARTSR)

Figure 1335. UART Mode Status Register (UARTSR)

Address: 0014H

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE3	PE2	PE1	PE0	RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF ⁽¹⁾ /RFE ⁽²⁾	DTF ⁽¹⁾ /TFE ⁽²⁾	NF
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C		W1C			W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read/cleared by software. Writing 1 clears contents.
2. Register bit is read-only field.

Table 1269. UARTSR field descriptions

Field	Description
0–15	Reserved Read returns 0
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCCR An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. <i>Note: For Baud rate above 1 Mbit/s, this flag is not usable.</i>

Table 1269. UARTSR field descriptions (continued)

Field	Description
18—21 PE[0:3]	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. 0 No parity error 1 Parity error in the corresponding received byte This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. <i>Note: Parity is checked only after a complete frame is received.</i> When WL bits = 10 or 11, either PE0 or PE2 is only set. – When PE0 is set, it indicates Parity error in either first or second byte received – When PE2 is set, it indicates Parity error in either third or fourth byte received
22 RMB	Release Message Buffer 0 Buffer data is free 1 Buffer data ready to be read by software This bit should be cleared by software. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.

Table 1269. UARTSR field descriptions (continued)

Field	Description
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software. The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode. An interrupt will be generated when LINIER.DBEIE/TOIE bit is set on the Error interrupt line in UART mode.
29 DRF/RFE	Data Reception Completed Flag /Rx FIFO Empty Flag DRF is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF bit is set only when the configured number of stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF is set irrespective of framing error, parity error or overrun error. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTF/TFF	Data Transmission Completed Flag/ TX FIFO Full Flag DTF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

56.4.2.7 LIN Time-Out Control Status Register (LINTCSR)

This register contains control and status bits for the timeout feature.

Figure 1336. LIN Time-Out Control Status Register (LINTCSR)

Address: 0018H

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MODE ⁽¹⁾	IOT ⁽¹⁾	TOCE	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.

Table 1270. LINTCSR field descriptions

Field	Description
0–20	Reserved Read returns 0.
21 MODE	Time-out counter mode 0 LIN mode 1 Output compare mode This bit can be configured only during initialization.
22 IOT	Idle on timeout 0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event
23 TOCE	Time-out counter enable 0 Time-out counter disable OCF flag is not set on an output compare event. 1 Time-out counter enable OCF flag is set if an output compare event occurs. TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE.
24–31 CNT[0:7]	Counter Value These bits reflect the value of a free-running counter used for timeout.

56.4.2.8 LIN Output Compare Register (LINOCR)

This register contains the value to be compared to the LINTOCR value. This register is writable by software only in Output Compare Mode.

Figure 1337. LIN Output Compare Register (LINOCR)

Address: 001CH

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OC27	OC26	OC25	OC24	OC23	OC22	OC21	OC20	OC17	OC16	OC15	OC14	OC13	OC12	OC11	OC10
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 1271. LINOCR field descriptions

Field	Description
0–15	Reserved Read returns 0.
16–23 OC2[7:0]	Output compare value 2
24–31 OC1[7:0]	Output compare value 1

56.4.2.9 LIN Time-Out Control Register (LINTOCR)

Figure 1338. LIN Time-Out Control Register (LINTOCR)

Address: 0020h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	RTO3	RTO2	RTO1	RTO0	0	HTO6	HTO5	HTO4	HTO3	HTO2	HTO1	HTO0
W																
Reset	0	0	0	0	1	1	1	0	0	0	0 ⁽¹⁾ /1	1 ⁽¹⁾ /0	1	1	0	0

1. These reset values are valid only for Master mode.

Table 1272. LINTOCR field descriptions

Field	Description
0–19	Reserved Read returns 0.
20–23 RTO[3:0]	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0Eh = 14, corresponding to $T_{\text{Response_Maximum}} = 1.4 \times T_{\text{Response_Nominal}}$
24	Reserved Read returns 0.
25–31 HTO[6:0]	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. The reset value is the 2Dh = 45, corresponding to $T_{\text{Header_Maximum}}$ in the LIN specification. If Master mode is enabled then these bits will always reflect 28 (1.4×10 bits of sync + 1.4×10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

56.4.2.10 LIN Fractional Baud Rate Register (LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

Note: When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

This register cannot be used when reduced oversampling is enabled (ROSE bit = 1)

Figure 1339. LIN Fractional Baud Rate Register (LINFBRR)

Address: 0024h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FBR3 ⁽¹⁾	FBR2 ⁽¹⁾	FBR1 ⁽¹⁾	FBR0 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.

Table 1273. LINFBR field descriptions

Field	Description												
0–27	Reserved Read returns 0												
28–31 FBR[3:0]	Fractional Baud rates <table border="1"> <thead> <tr> <th>LINFBR[4:0]</th><th>Fraction (LDIV)</th></tr> </thead> <tbody> <tr> <td>0h</td><td>0</td></tr> <tr> <td>1h</td><td>1/16</td></tr> <tr> <td>...</td><td>...</td></tr> <tr> <td>Eh</td><td>14/16</td></tr> <tr> <td>Fh</td><td>15/16</td></tr> </tbody> </table>	LINFBR[4:0]	Fraction (LDIV)	0h	0	1h	1/16	...	...	Eh	14/16	Fh	15/16
LINFBR[4:0]	Fraction (LDIV)												
0h	0												
1h	1/16												
...	...												
Eh	14/16												
Fh	15/16												

56.4.2.11 LIN Integer Baud Rate Register (LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBR. It can be programmed only in Initialization mode.

Note: When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Figure 1340. LIN Integer Baud Rate Register (LINIBRR)

Address : 0028h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	IBR19 ⁽¹⁾	IBR18 ⁽¹⁾	IBR17 ⁽¹⁾	IBR16 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IBR15 ⁽¹⁾	IBR14 ⁽¹⁾	IBR13 ⁽¹⁾	IBR12 ⁽¹⁾	IBR11 ⁽¹⁾	IBR10 ⁽¹⁾	IBR9 ⁽¹⁾	IBR8 ⁽¹⁾	IBR7 ⁽¹⁾	IBR6 ⁽¹⁾	IBR5 ⁽¹⁾	IBR4 ⁽¹⁾	IBR3 ⁽¹⁾	IBR2 ⁽¹⁾	IBR1 ⁽¹⁾	IBR0 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.

Table 1274. LINIBRR field descriptions

Field	Description												
0–11	Reserved Read returns 0.												
12–31 IBR[19:0]	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. <table border="1"> <thead> <tr> <th>LINIBRR[19:0]</th><th>Mantissa (LDIV)</th></tr> </thead> <tbody> <tr> <td>00000h</td><td>LIN clock disabled</td></tr> <tr> <td>00001h</td><td>1</td></tr> <tr> <td>...</td><td>...</td></tr> <tr> <td>FFFFEh</td><td>1048574</td></tr> <tr> <td>FFFFFh</td><td>1048575</td></tr> </tbody> </table>	LINIBRR[19:0]	Mantissa (LDIV)	00000h	LIN clock disabled	00001h	1	...	...	FFFFEh	1048574	FFFFFh	1048575
LINIBRR[19:0]	Mantissa (LDIV)												
00000h	LIN clock disabled												
00001h	1												
...	...												
FFFFEh	1048574												
FFFFFh	1048575												

56.4.2.12 LIN Checksum Field Register (LINCFR)

Note: There is a delay between 4 to 6 clock cycles of PBRIDGE_x_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.

This register consists of checksum bits.

Figure 1341. LIN Checksum Field Register (LINCFR)

Address: 002Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1275. LINCFR field descriptions

Field	Description																				
0–23	Reserved Read returns 0.																				
24–31 CF[7:0]	Checksum bits When the CCD bit is reset these bits are read-only and are calculated by hardware. When the CCD bit is set, these bits can be written by software. <table><tr><th>CFD</th><th>CCD</th><th>LINCHKSUM Read/write</th><th>Checksum sent</th></tr><tr><td>1</td><td>1</td><td>read/write</td><td>None</td></tr><tr><td>1</td><td>0</td><td>read-only</td><td>None</td></tr><tr><td>0</td><td>1</td><td>read/write</td><td>Programmed checksum</td></tr><tr><td>0</td><td>0</td><td>read-only</td><td>Calculated checksum</td></tr></table>	CFD	CCD	LINCHKSUM Read/write	Checksum sent	1	1	read/write	None	1	0	read-only	None	0	1	read/write	Programmed checksum	0	0	read-only	Calculated checksum
CFD	CCD	LINCHKSUM Read/write	Checksum sent																		
1	1	read/write	None																		
1	0	read-only	None																		
0	1	read/write	Programmed checksum																		
0	0	read-only	Calculated checksum																		

56.4.2.13 LIN Control Register 2 (LINC2)

This register includes control status bits related to buffer operations.

Figure 1342. LIN Control Register 2 (LINC2)

Address : 0030h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBDE ⁽¹⁾	IOBE ⁽¹⁾	IOPE ⁽¹⁾	WURQ ⁽²⁾	DDRQ ⁽²⁾	DTRQ ⁽²⁾	ABRQ ⁽²⁾	HTRQ ⁽²⁾	0	0	0	0	0	0	0	0
W																
Reset	0	1	0 ⁽³⁾ /1	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.
2. Register bit can be read/set by software.
3. When slave = 0, then this bit will always read a 0, and cannot be programmed.

Table 1276. LINC2 field descriptions

Field	Description
0–15	Reserved Read returns 0.
16 TBDE	Two Bit delimiter bit 0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits This bit can be set in Initialization mode only.
17 IOBE	Idle on Bit Error 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine This bit can be set in Initialization mode only.
18 IOPE	Idle on Identifier Parity Error 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine This bit can be set in Initialization mode only. If generic <i>slave</i> = 0, then this bit will always read a 0, and cannot be programmed.
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request.
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered.
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete.
22 ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also.

Table 1276. LINCR2 field descriptions (continued)

Field	Description
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header. Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. <i>Note: In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.</i>
24–31	Reserved Read returns 0.

56.4.2.14 Buffer Identifier Register (BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

Note: All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Figure 1343. Buffer Identifier Register (BIDR)

Address: 0034h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL5	DFL4	DFL3	DFL2	DFL1	DFL0	DIR	CCS	0	0	ID5	ID4	ID3	ID2	ID1	ID0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1277. BIDR field descriptions

Field	Description
0–15	Reserved Read returns 0.
16–21 DFL[5:0]	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes - 1 <i>Note: DFL[5:3] is provided for extended frames. Normally LIN mode will use DFL[2:0]. Identifier filters are now compatible with DFL[5:0] also.</i>

Table 1277. BDR field descriptions (continued)

Field	Description
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copy them in the BDR registers 1 LINFlexD transmits the data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only This is compatible with LIN specification 1.3 and lower.
24–25	Reserved Forced to 0 by hardware.
26–31 ID[5:0]	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = '1').

56.4.2.15 Buffer Data Register Least Significant (BDRL)

This register is a part of an 8-byte data buffer.

Figure 1344. Buffer Data Register Least Significant (BDRL)

Address: 0038h

Access: User read/write

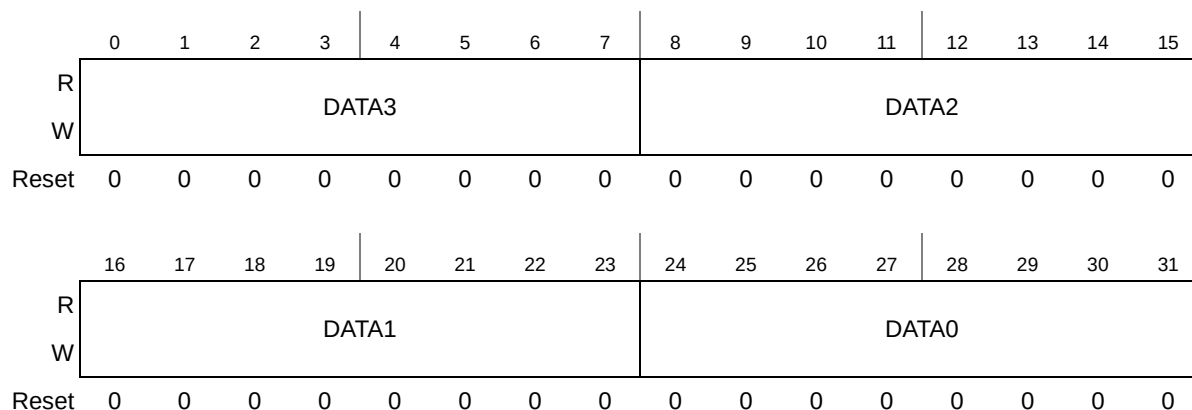


Table 1278. BDRL field descriptions

Field	Description
0–7 DATA3[7:0]	Data Byte 3 Data byte 3 of the data field.
8–15 DATA2[7:0]	Data Byte 2 Data byte 2 of the data field.

Table 1278. BDRL field descriptions (continued)

Field	Description
16–23 DATA1[7:0]	Data Byte 1 Data byte 1 of the data field.
24–31 DATA0[7:0]	Data Byte 0 Data byte 0 of the data field.

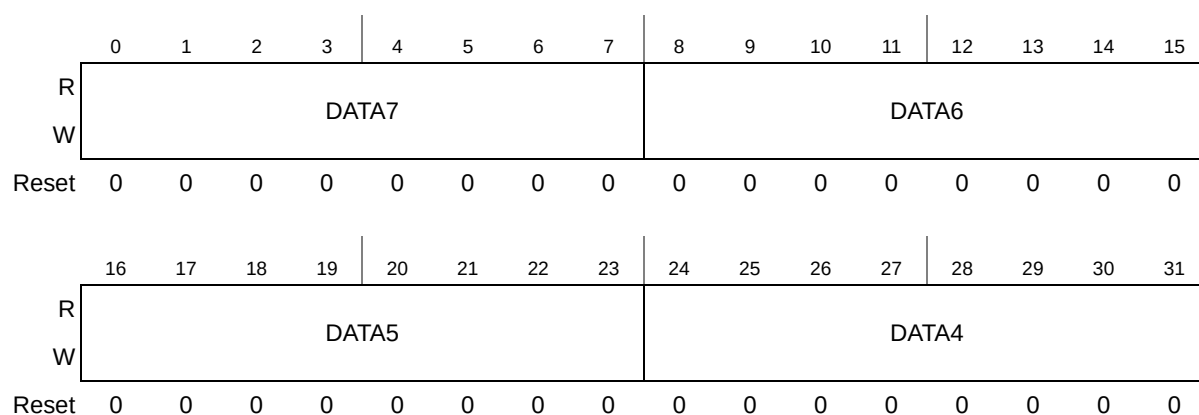
56.4.2.16 Buffer Data Register Most Significant (BDRM)

This register is a part of an 8-byte data buffer.

Figure 1345. Buffer Data Register Most Significant (BDRM)

Address: 003Ch

Access: User read/write

**Table 1279. BDRM field descriptions**

Field	Description
0–7 DATA7[7:0]	Data Byte 7 Data byte 7 of the data field.
8–15 DATA6[7:0]	Data Byte 6 Data byte 6 of the data field.
16–23 DATA5[7:0]	Data Byte 5 Data byte 5 of the data field.
24–31 DATA4[7:0]	Data Byte 4 Data byte 4 of the data field.

56.4.2.17 Identifier Filter Enable Register (IFER)

This register enables/disables a particular filter.

Figure 1346. Identifier Filter Enable Register (IFER)

Address: 0040h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FACT15 ⁽¹⁾	FACT14 ⁽¹⁾	FACT13 ⁽¹⁾	FACT12 ⁽¹⁾	FACT11 ⁽¹⁾	FACT10 ⁽¹⁾	FACT9 ⁽¹⁾	FACT8 ⁽¹⁾	FACT7 ⁽¹⁾	FACT6 ⁽¹⁾	FACT5 ⁽¹⁾	FACT4 ⁽¹⁾	FACT3 ⁽¹⁾	FACT2 ⁽¹⁾	FACT1 ⁽¹⁾	FACT0 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.

Table 1280. IFER field descriptions

Field	Description
0–15	Reserved Read returns 0
16–31 FACT[15:0]	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. 1. In Identifier mask mode, FACT (2n+1) have no effect on the corresponding filters as they act as mask for the Identifier 2n. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. <i>Note: Refer to device configuration chapter to see the number of filters used in the device.</i>

56.4.2.18 Identifier Filter Match Index (IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Figure 1347. Identifier Filter Match Index (IFMI)

Address: 0044h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	IFM14	IFM13	IFM12	IFM11	IFM10
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1281. IFMI field descriptions

Field	Description
0–26	Reserved Read returns 0.
27–31 IFMI[4:0]	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see Section 56.3.2.4, Slave mode for more details).

56.4.2.19 Identifier Filter Mode Register (IFMR)

This register configures the modes of filters.

Figure 1348. Identifier Filter Mode Register (IFMR)

Address: 0048h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	IFM7 ⁽¹⁾	IFM6 ⁽¹⁾	IFM5 ⁽¹⁾	IFM4 ⁽¹⁾	IFM3 ⁽¹⁾	IFM2 ⁽¹⁾	IFM1 ⁽¹⁾	IFM0 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.

Table 1282. IFMR field descriptions

Field	Description
0–23	Reserved Read returns 0.
24–31 IFM[7:0]	Filter mode 0 Filters 2n and 2n+1 are in identifier list mode 1 Filters 2n and 2n+1 are in mask mode, where filter 2n+1 is the mask for filter 2n

56.4.2.20 Identifier Filter Control Register (IFCR2n)

This register is read-only in normal mode and can be programmed only in Initialization mode. This register acts as a filter in both Identifier list and Identifier mask modes.

Figure 1349. Identifier Filter Control Register (IFCR2n)

Address: $(4C+8 \times n)$ where $n = 0$ to $\text{no_of_filters}^2/2-1$

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL5 ⁽¹⁾	DFL4 ⁽¹⁾	DFL3 ⁽¹⁾	DFL2 ⁽¹⁾	DFL1 ⁽¹⁾	DFL0 ⁽¹⁾	DIR ⁽¹⁾	CCS ⁽¹⁾	0	0	ID5 ⁽¹⁾	ID4 ⁽¹⁾	ID3 ⁽¹⁾	ID2 ⁽¹⁾	ID1 ⁽¹⁾	ID0 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.
2. Refer to device configuration chapter to see the number of filters used in the device

Table 1283. IFCR2n field descriptions

Field	Description
0–15	Reserved Read returns a 0.
16–21 DFL[5:0]	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes – 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only This is compatible with LIN specification 1.3 and lower.
24–25	Reserved Forced to 0 by hardware.
26–31 ID[5:0]	Identifier Identifier part of the identifier field without the identifier parity.

56.4.2.21 Identifier Filter Control Register (IFCR2n+1)

This register is read-only in normal mode and can be programmed only in Initialization mode. This register acts as a filter in Identifier list mode but acts as a mask for IFCR2n in Identifier mask mode.

Figure 1350. Identifier Filter Control Register (IFCR2n+1)

Address: $(50+8 \times n)$ where $n = 0$ to $\text{no_of_filters}^{(2)}/2-1$

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL5 ⁽¹⁾	DFL4 ⁽¹⁾	DFL3 ⁽¹⁾	DFL2 ⁽¹⁾	DFL1 ⁽¹⁾	DFL0 ⁽¹⁾	DIR ⁽¹⁾	CCS ⁽¹⁾	0	0	ID5 ⁽¹⁾	ID4 ⁽¹⁾	ID3 ⁽¹⁾	ID2 ⁽¹⁾	ID1 ⁽¹⁾	ID0 ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.
2. Refer to device configuration chapter to see the number of filters used in the device

Table 1284. IFCR2n+1 field descriptions

Field	Description
0–15	Reserved Read returns a 0
16–21 DFL[5:0]	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes – 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only This is compatible with LIN specification 1.3 and lower.
24–25	Reserved Forced to 0 by hardware.
26–31 ID[5:0]	Identifier Identifier part of the identifier field without the identifier parity.

56.4.2.22 Global Control Register (GCR)

This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

Figure 1351. Global Control Register (GCR)

Address: $4C + 4 \times \text{no_of_filters} \times \text{slave}^{(3)}$

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDFBM ⁽¹⁾	RDFBM ⁽¹⁾	TDLIS ⁽¹⁾	RDLIS ⁽¹⁾	STOP ⁽¹⁾	SR ⁽²⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in Initialization mode.
2. Register bit can be written only by software in Initialization mode. Bit is always read 0 by software.
3. Refer to device configuration chapter to see the number of filters and value of slave used in the device.

Table 1285. GCR field descriptions

Field	Description
0–25	Reserved Forced to 0 by hardware
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. 0 The first bit of received data is LSB — in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB — in other words, first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. 0 Transmitted data is not inverted 1 Transmitted data is inverted

Table 1285. GCR field descriptions (continued)

Field	Description														
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. 0 Received data is not inverted 1 Received data is inverted														
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). 0 1 stop bit 1 2 stop bits														
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). This bit is always read as 0. The following table describes, the register fields reset by SR: <table> <tr> <th>Register</th><th>Comment</th></tr> <tr> <td>LINSR</td><td>All fields except RXbusy and AUTOSYNC_COMP are reset</td></tr> <tr> <td>LINESR</td><td>All fields are reset</td></tr> <tr> <td>LINTCSR</td><td>Only CNT[0:7] is reset</td></tr> <tr> <td>UARTSR</td><td>All fields except RFE & TFF are reset</td></tr> <tr> <td>UARTCR</td><td>Only TFC & RFC are reset</td></tr> <tr> <td>UARTCTO</td><td>All fields are reset</td></tr> </table>	Register	Comment	LINSR	All fields except RXbusy and AUTOSYNC_COMP are reset	LINESR	All fields are reset	LINTCSR	Only CNT[0:7] is reset	UARTSR	All fields except RFE & TFF are reset	UARTCR	Only TFC & RFC are reset	UARTCTO	All fields are reset
Register	Comment														
LINSR	All fields except RXbusy and AUTOSYNC_COMP are reset														
LINESR	All fields are reset														
LINTCSR	Only CNT[0:7] is reset														
UARTSR	All fields except RFE & TFF are reset														
UARTCR	Only TFC & RFC are reset														
UARTCTO	All fields are reset														

56.4.2.23 UART Preset Timeout Register (UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be received or to monitor the idle state of the reception line. This register can be written by software any time.

Figure 1352. UART Preset Timeout Register (UARTPTO)

Address: (GCR address)⁽¹⁾+04h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PTO11	PTO10	PTO9	PTO8	PTO7	PTO6	PTO5	PTO4	PTO3	PTO2	PTO1	PTO0
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

1. Depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Table 1286. UARTPTO field descriptions

Field	Description
0–19	Reserved Read returns 0.
20–31 PTO[11:0]	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer also to register UARTCTO.

56.4.2.24 UART Current Timeout Register (UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see [Section 56.4.2.23, UART Preset Timeout Register \(UARTPTO\)](#)) to monitor the number of bits received by UART or to monitor the idle state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGE_{Ex}_CLK (when ROSE = 0)
- Is clocked with LIN_CLK / (OSR * IBRR) synchronized to PBRIDGE_{Ex}_CLK (when ROSE = 1)
- Is automatically enabled when UARTCR[RXEN] = 1

Note: Due to this synchronization there is a delay between 4 to 6 clock cycles of PBRIDGE_{Ex}_CLK for the internal counter's value (which is clocked with LIN_CLK/16 * LDIV when ROSE = 0 or LIN_CLK/OSR * IBRR when ROSE = 1) to reflect on UARTCTO.

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO
- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

Figure 1353. UART Current Timeout Register (UARTCTO)

Address: (GCR address)⁽¹⁾+08h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CTO11	CTO10	CTO9	CTO8	CTO7	CTO6	CTO5	CTO4	CTO3	CTO2	CTO1	CTO0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Table 1287. UARTCTO field descriptions

Field	Description
0–19	Reserved Read returns 0.
20–31 CTO[11:0]	Current Timeout CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

56.4.2.25 DMA Tx Enable Register (DMATXE)

This register enables the DMA TX interface. This register can be written and read by software any time.

Figure 1354. DMA Tx Enable Register (DMATXE)

Address: (GCR address)⁽¹⁾+0Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTE15	DTE14	DTE13	DTE12	DTE11	DTE10	DTE9	DTE8	DTE7	DTE6	DTE5	DTE4	DTE3	DTE2	DTE1	DTE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Table 1288. DMATXE field descriptions

Field	Description
0 – (2**TX_CH_NUM–1)	Reserved Read returns 0. <i>Note: Refer to the device configuration chapter for the value of TX_CH_NUM used in this device.</i>
(2**TX_CH_NUM) – 31 DTE[2**TX_CH_NUM–1:0] ⁽¹⁾	DMA Tx channel Y enable 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled <i>Note: The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state.</i> <i>Note: Refer to the device configuration chapter for the value of TX_CH_NUM used in this device.</i>

1. ** stands for exponentiation.

56.4.2.26 DMA Rx Enable Register (DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

Figure 1355. DMA Rx Enable Register (DMARXE)

Address: (GCR address)⁽¹⁾+10h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRE15	DRE14	DRE13	DRE12	DRE11	DRE10	DRE9	DRE8	DRE7	DRE6	DRE5	DRE4	DRE3	DRE2	DRE1	DRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Table 1289. DMARXE field descriptions

Field	Description
0 – (2**RX_CH_NUM–1)	Reserved Read returns 0. <i>Note: Refer to the device configuration chapter for the value of RX_CH_NUM used in this device.</i>
(2**RX_CH_NUM) – 31 DRE[2**RX_CH_NUM–1:0] ⁽¹⁾	DMA Rx channel Y enable 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled <i>Note: The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state.</i> <i>Note: Refer to the device configuration chapter for the value of RX_CH_NUM used in this device.</i>

1. ** stands for exponentiation.



56.4.3 Register map

56.4.3.1 IPS bus

Table 1290. LINFlexD register map

Addr offset	Register name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	LINCRI	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		CCD	CFD	LASE	AUTO WU	MBL3	MBL2	MBL1	MBL0	BF	SLFM	LBKM	MME	SSBL	RBLM	SLEEP	INIT
4	LINIER	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		SZIE	OCIE	BEIE	CEIE	HEIE	—	—	FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIE /TOIE	DRIE	DTIE	HRIE
8	LINSR	—	—	—	—	—	—	—	—	—	—	—	—	AUTO SYNC _COMP	RDC		
		LINS3	LINS2	LINS1	LINS0	—	—	RMB	DRBN E	RXbus y	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
C	LINESR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		SZF	OCF	BEF	CEF	SFEF	SDEF	IDPEF	FEF	BOF	—	—	—	—	—	—	NF
10	UARTCR	MIS	CSP			OSR				ROSE	NEF			DTU	SBUR		WLS
		TDFL/TFC			RDFL/RFC			RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE	WL0	UART
14	UARTSR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		SZF	OCF	PE3	PE2	PE1	PE0	RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF/R FE	DTF/T FF	NF
18	LINTCSR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	MODE	IOT	TOCE	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
1C	LINOCCR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		OC27	OC26	OC25	OC24	OC23	OC22	OC21	OC20	OC17	OC16	OC15	OC14	OC13	OC12	OC11	OC10



Table 1290. LINFlexD register map

Addr offset	Register name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
20	LINTOCR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	RTO3	RTO2	RTO1	RTO0	—	HTO6	HTO5	HTO4	HTO3	HTO2	HTO1	HTO0
24	LINFBR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	—	—	—	—	FBR3	FBR2	FBR1	FBR0
28	LINIBRR	—	—	—	—	—	—	—	—	—	—	—	—	IBR 19	IBR 18	IBR 17	IBR 16
		IBR 15	IBR 14	IBR 13	IBR 12	IBR 11	IBR 10	IBR9	IBR8	IBR7	IBR6	IBR5	IBR4	IBR3	IBR2	IBR1	IBR0
2C	LINCFR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
30	LINC2R	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		TBDE	IOBE	IOPE	WUR Q	DDRQ	DTRQ	ABRQ	HTRQ	—	—	—	—	—	—	—	—
34	BIDR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DFL5	DFL4	DFL3	DFL2	DFL1	DFL0	DIR	CCS	—	—	ID5	ID4	ID3	ID2	ID1	ID0
38	BDRL	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2
		DATA1	DATA1	DATA1	DATA1	DATA1	DATA1	DATA1	DATA1	DATA0	DATA0	DATA0	DATA0	DATA0	DATA0	DATA0	DATA0
3C	BDRM	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6
		DATA5	DATA5	DATA5	DATA5	DATA5	DATA5	DATA5	DATA5	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4
40	IFER	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		FACT 15	FACT 14	FACT 13	FACT 12	FACT 11	FACT 10	FACT 9	FACT 8	FACT 7	FACT 6	FACT 5	FACT 4	FACT 3	FACT 2	FACT 1	FACT 0
44	IFMI	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	—	—	—	IFMI4	IFMI3	IFMI2	IFMI1	IFMI0
48	IFMR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	IFM7	IFM6	IFM5	IFM4	IFM3	IFM2	IFM1	IFM0

Table 1290. LINFlexD register map

Addr offset	Register name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4C+ 8×n	IFCR2n	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DFL5	DFL4	DFL3	DFL2	DFL1	DFL0	DIR	CCS	—	—	ID5	ID4	ID3	ID2	ID1	ID0
50+ 8×n	IFCR2n+1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DFL5	DFL4	DFL3	DFL2	DFL1	DFL0	DIR	CCS	—	—	ID5	ID4	ID3	ID2	ID1	ID0
4C+8 ×n+04 h	GCR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	—	—	TDFB M	RDFB M	TDLIS	RDLIS	STOP	SR
GCR offset +04h	UART PTO	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	PTO 11	PTO 10	PTO9	PTO8	PTO7	PTO6	PTO5	PTO4	PTO3	PTO2	PTO1	PTO0
GCR offset +08h	UART CTO	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	CTO1 1	CTO1 0	CTO9	CTO8	CTO7	CTO6	CTO5	CTO4	CTO3	CTO2	CTO1	CTO0
GCR offset +0Ch	DMATXE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DTE 15	DTE 14	DTE 13	DTE 12	DTE 11	DTE 10	DTE 9	DTE8	DTE7	DTE6	DTE5	DTE4	DTE3	DTE2	DTE1	DTE0
GCR offset +10h	DMARXE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DRE 15	DRE 14	DRE 13	DRE 12	DRE 11	DRE 10	DRE 9	DRE8	DRE7	DRE6	DRE5	DRE4	DRE3	DRE2	DRE1	DRE0

56.5 Programming considerations

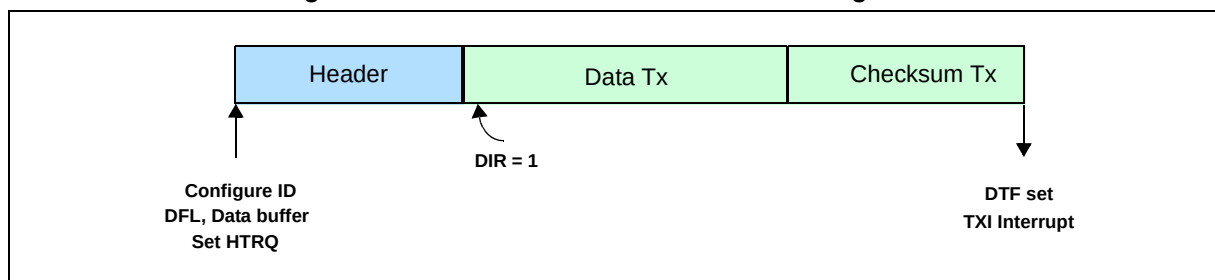
The next subsections describe the various configurations in which the LINFlexD module can be used.

56.5.1 Master node

LINFlexD acts as Master when the MME bit is set.

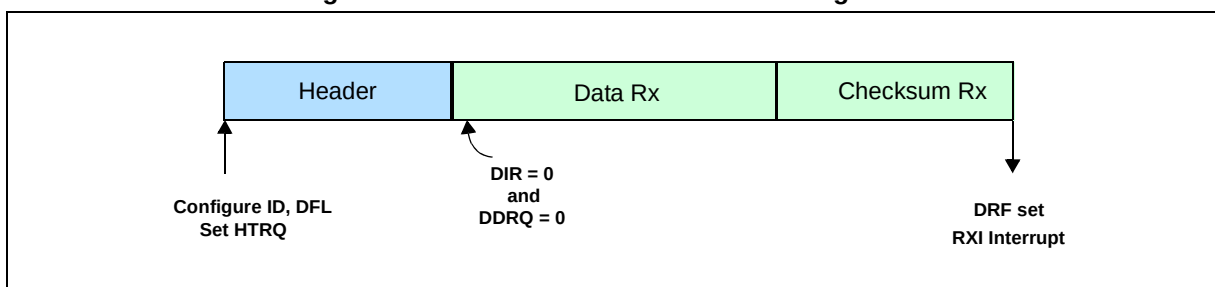
56.5.1.1 Transmitter

Figure 1356. Master node — transmitter configuration



56.5.1.2 Receiver

Figure 1357. Master node — receiver configuration



56.5.1.3 Transmitter, Bit Error

Figure 1358. Master node — transmitter, bit error configuration

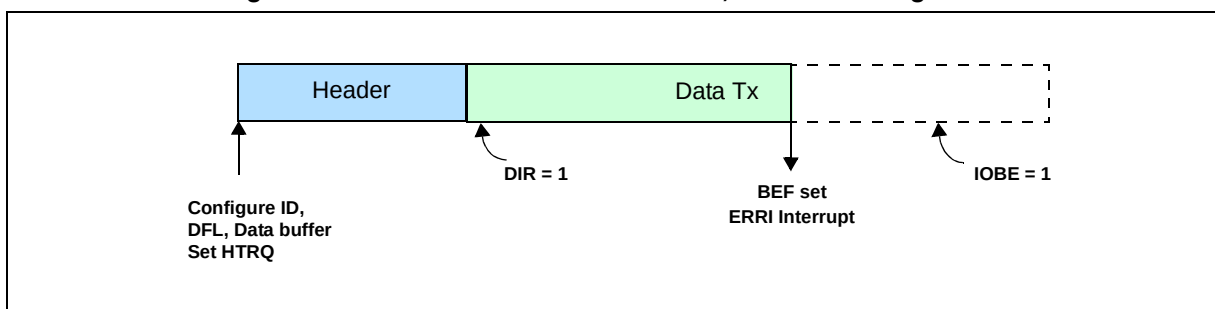
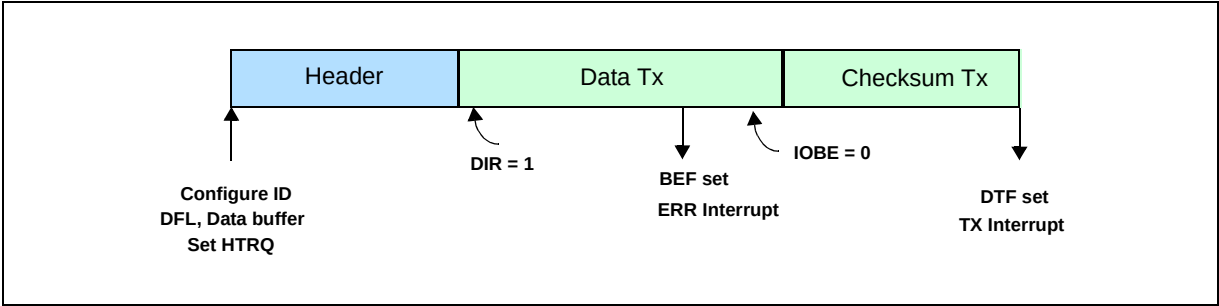
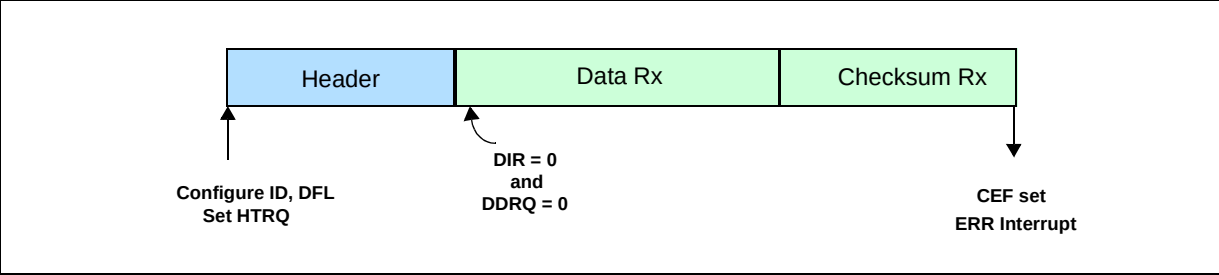


Figure 1359. Master node — transmitter, checksum error configuration



56.5.1.4 Receiver, Checksum Error

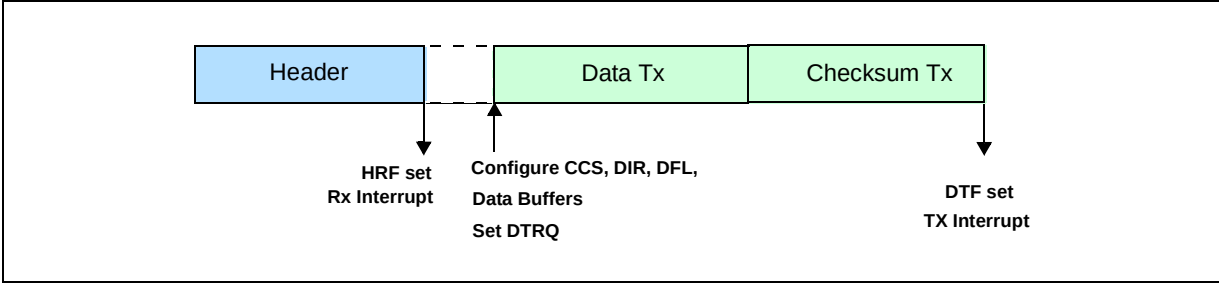
Figure 1360. Master node — receiver, checksum error configuration



56.5.2 Slave node

56.5.2.1 Transmitter (no identifier filters)

Figure 1361. Slave node — transmitter (no identifier filters) configuration



56.5.2.2 Receiver (no identifier filters)

Figure 1362. Slave node — receiver (no identifier filters) configuration

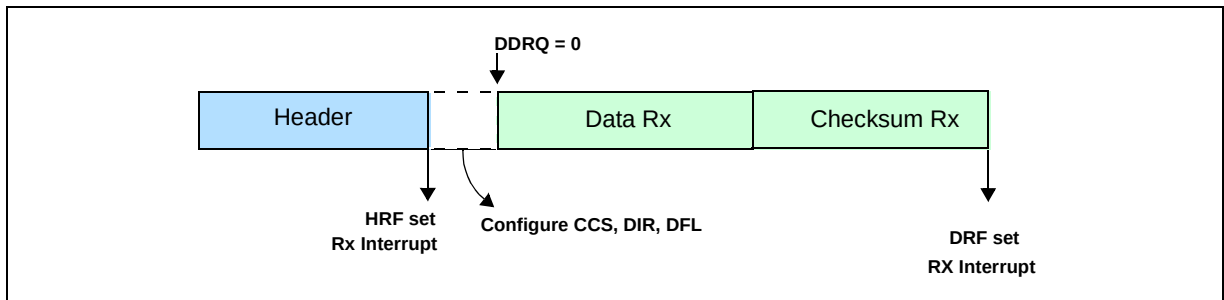
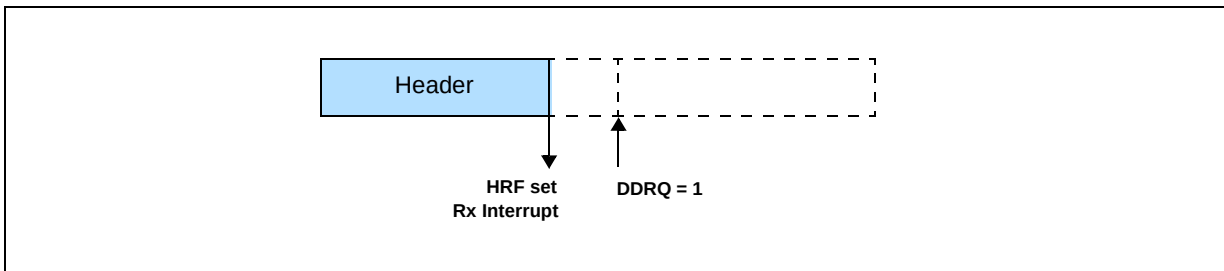
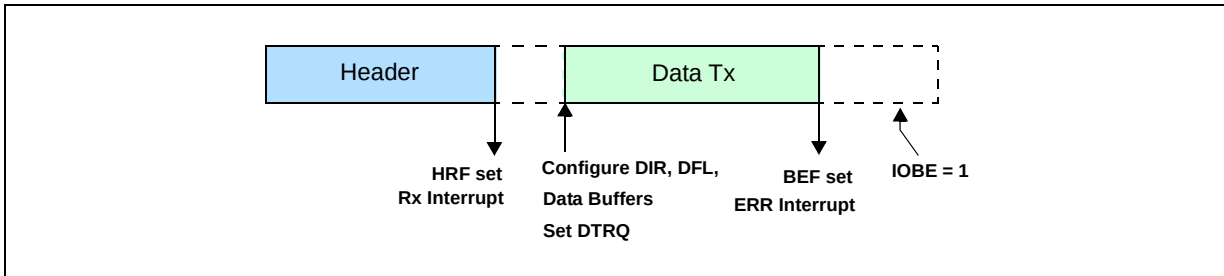


Figure 1363. Slave node — receiver (no identifier filters) configuration



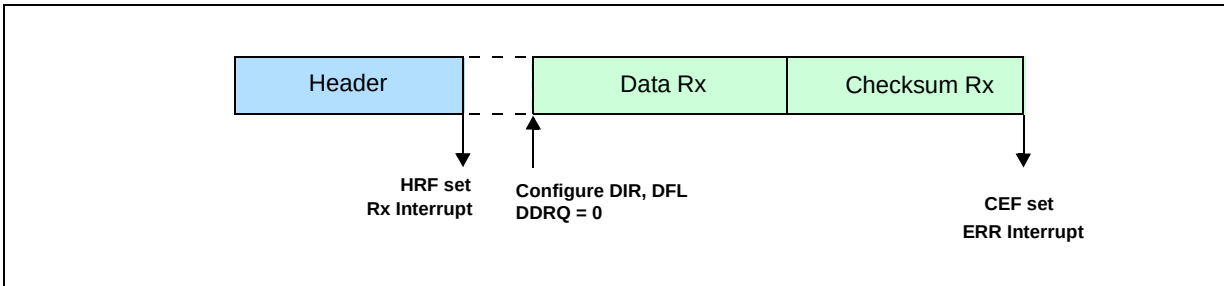
56.5.2.3 No filters, transmitter, bit error

Figure 1364. Slave node — No filters, transmitter, bit error configuration



56.5.2.4 No filters, receiver, checksum error

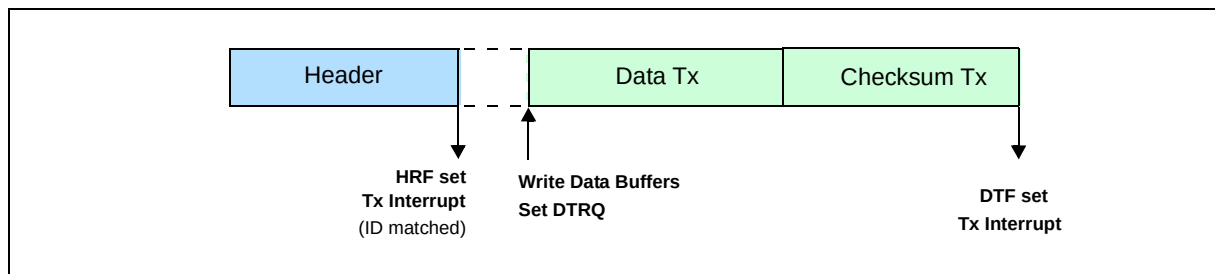
Figure 1365. Slave node — No filters, receiver, checksum error configuration



56.5.2.5 At least one TX filter, BF is reset, ID matches filter

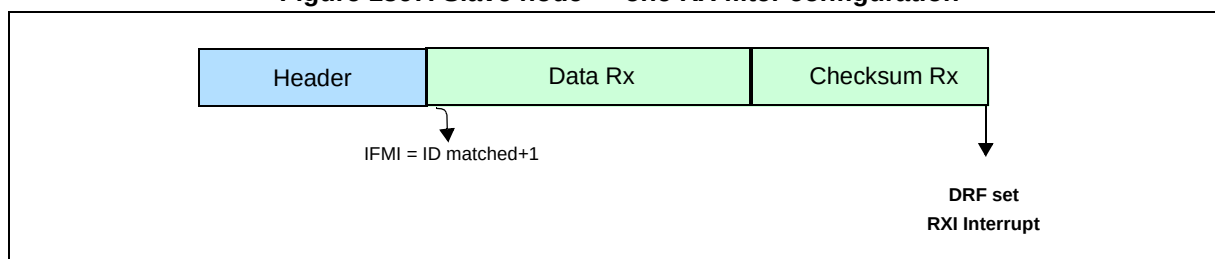
This configuration can be used in case slave never receives data, for example, sensor.

Figure 1366. Slave node — one TX filter configuration



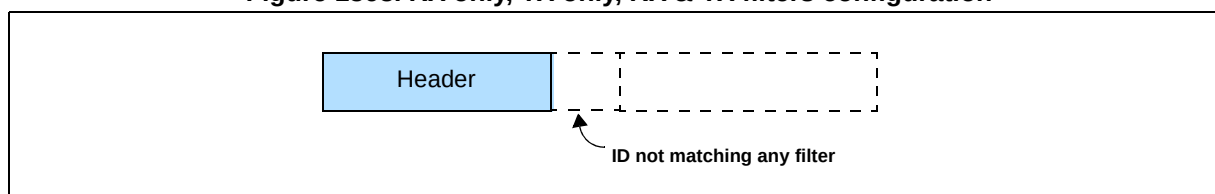
56.5.2.6 At least one RX filter, BF reset, ID matches filter

Figure 1367. Slave node — one RX filter configuration



56.5.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

Figure 1368. RX only, TX only, RX & TX filters configuration



56.5.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

Figure 1369. TX filter, BF is set configuration — ID has matched

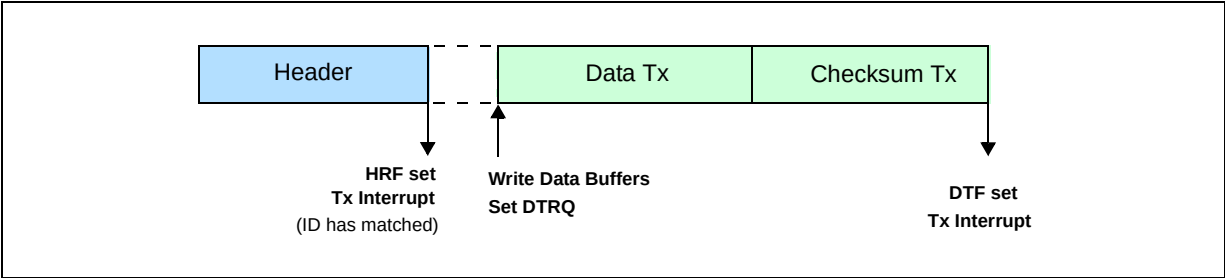
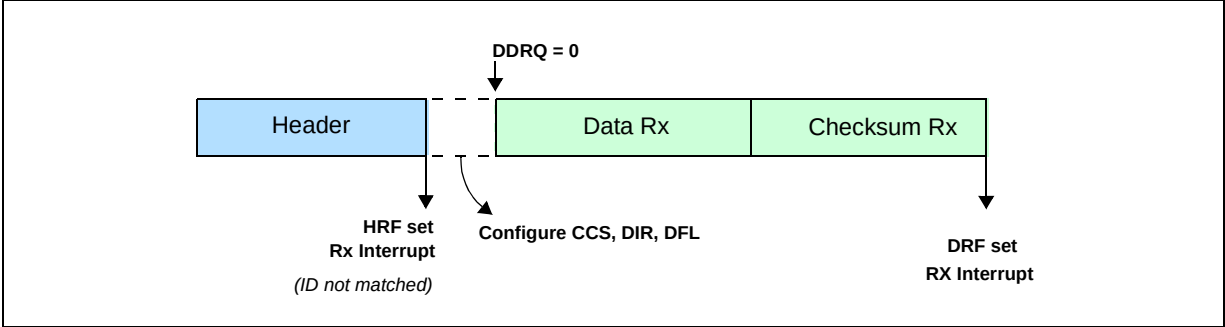


Figure 1370. TX filter, BF is set configuration — ID not matched



56.5.2.9 RX filter, BF is set

Figure 1371. RX filter, BF is set configuration — ID matched

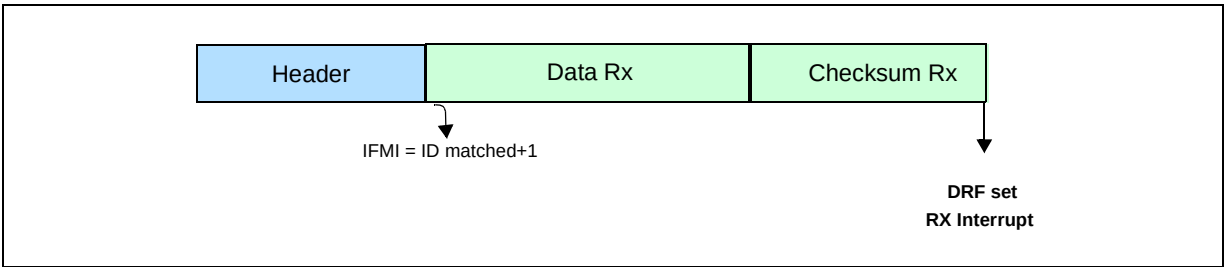


Figure 1372. RX filter, BF is set configuration — ID not matched (ID is Rx)

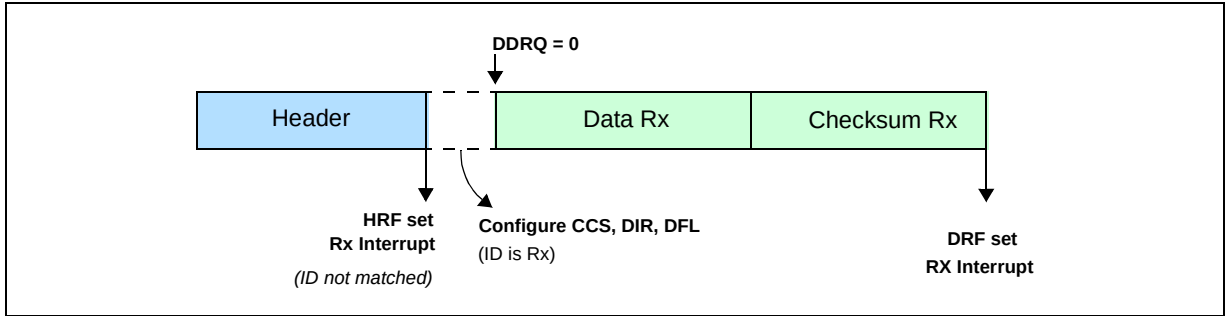
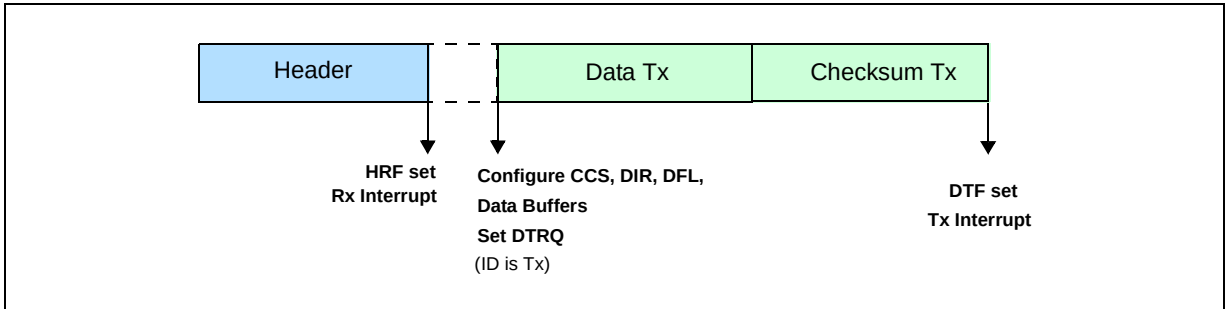


Figure 1373. RX filter, BF is set configuration — (ID is Tx)



56.5.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

Figure 1374. TX filter, RX filter, BF set configuration — (ID matched)

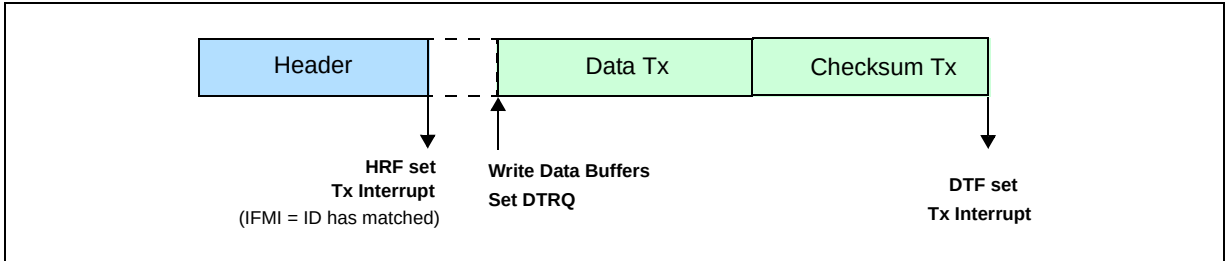


Figure 1375. TX filter, RX filter, BF set configuration — (ID matched + 1)

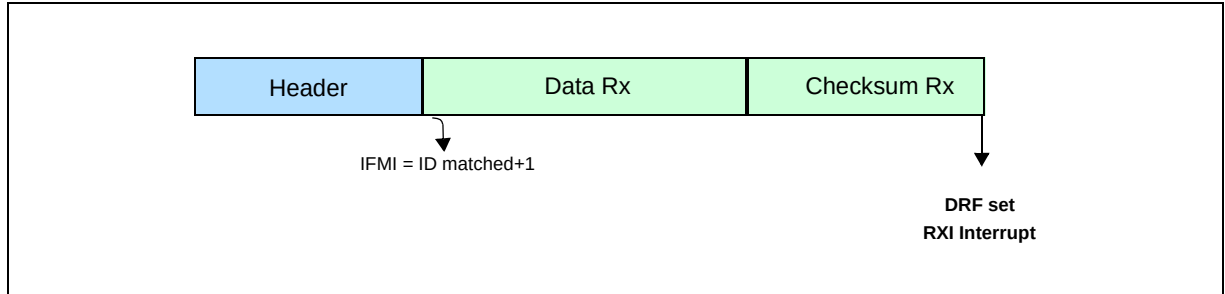
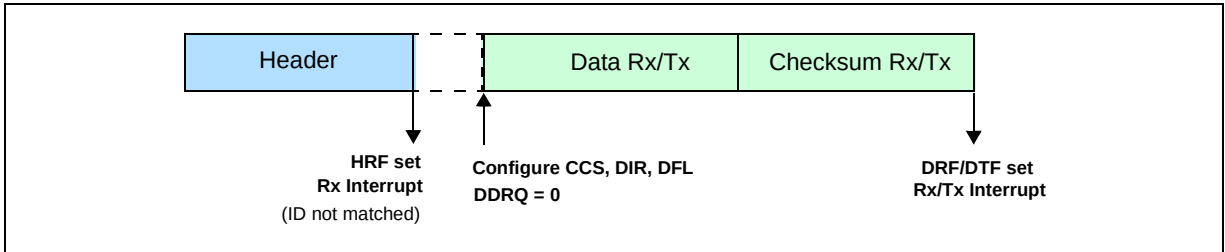


Figure 1376. TX filter, RX filter, BF set configuration — (ID not matched, RX/TX Interrupt)



56.5.3 Extended frames

Figure 1377. Extended frames (TX Interrupt)

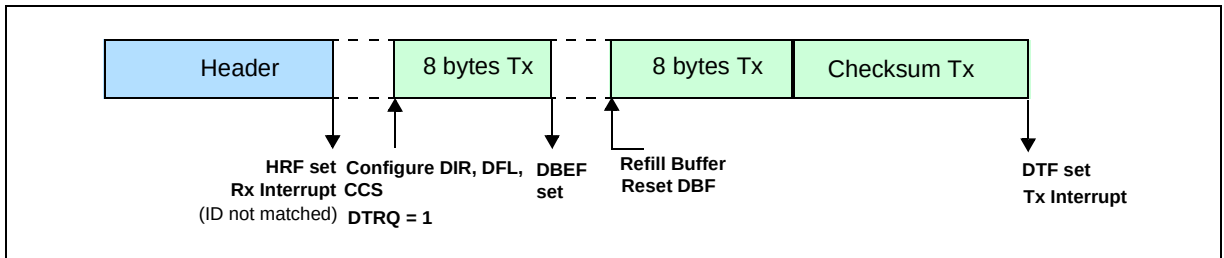
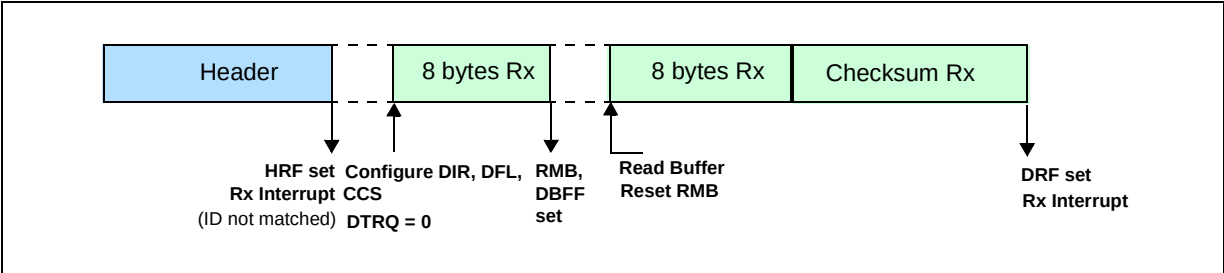


Figure 1378. Extended frames (RX Interrupt)



56.5.4 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

Figure 1379. Response timeout

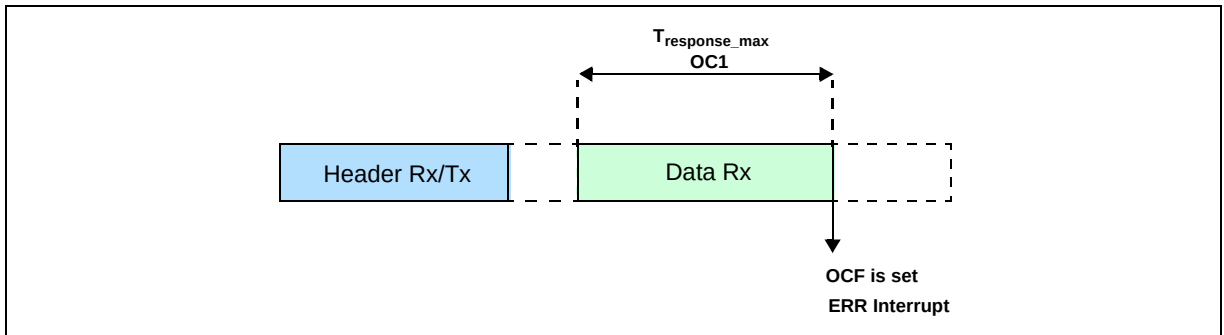


Figure 1380. Frame timeout

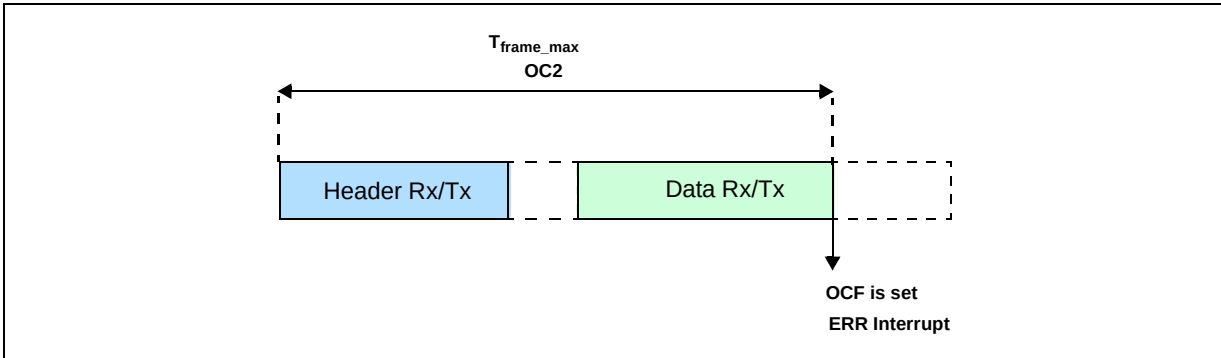
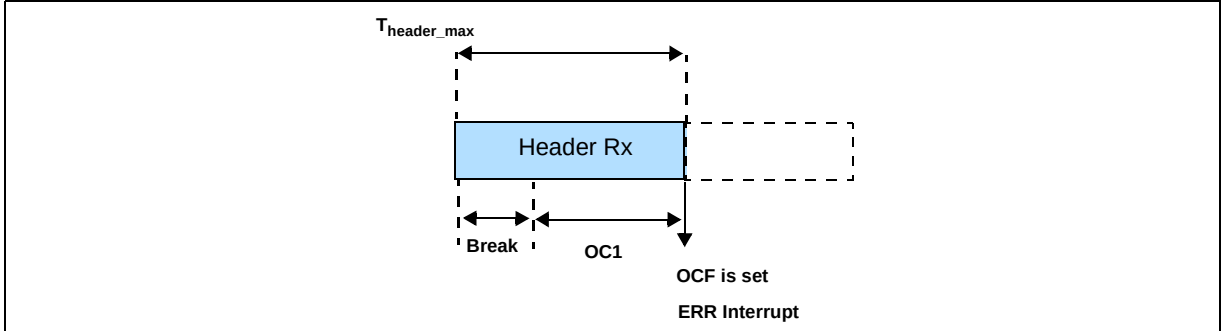
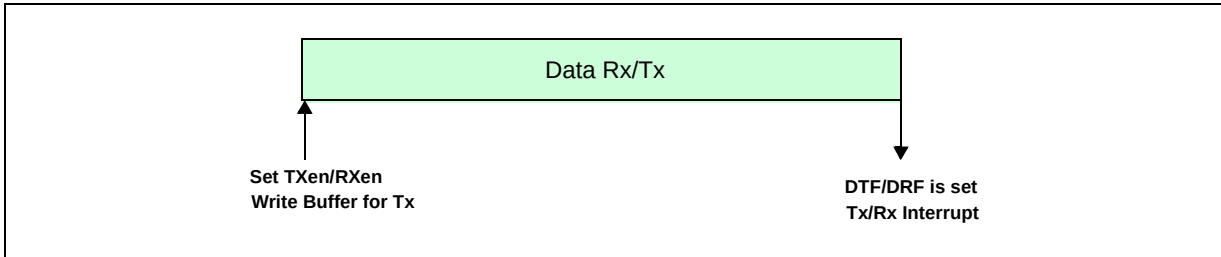


Figure 1381. Header timeout



56.5.5 UART mode

Figure 1382. UART mode configuration



56.5.6 Interrupts

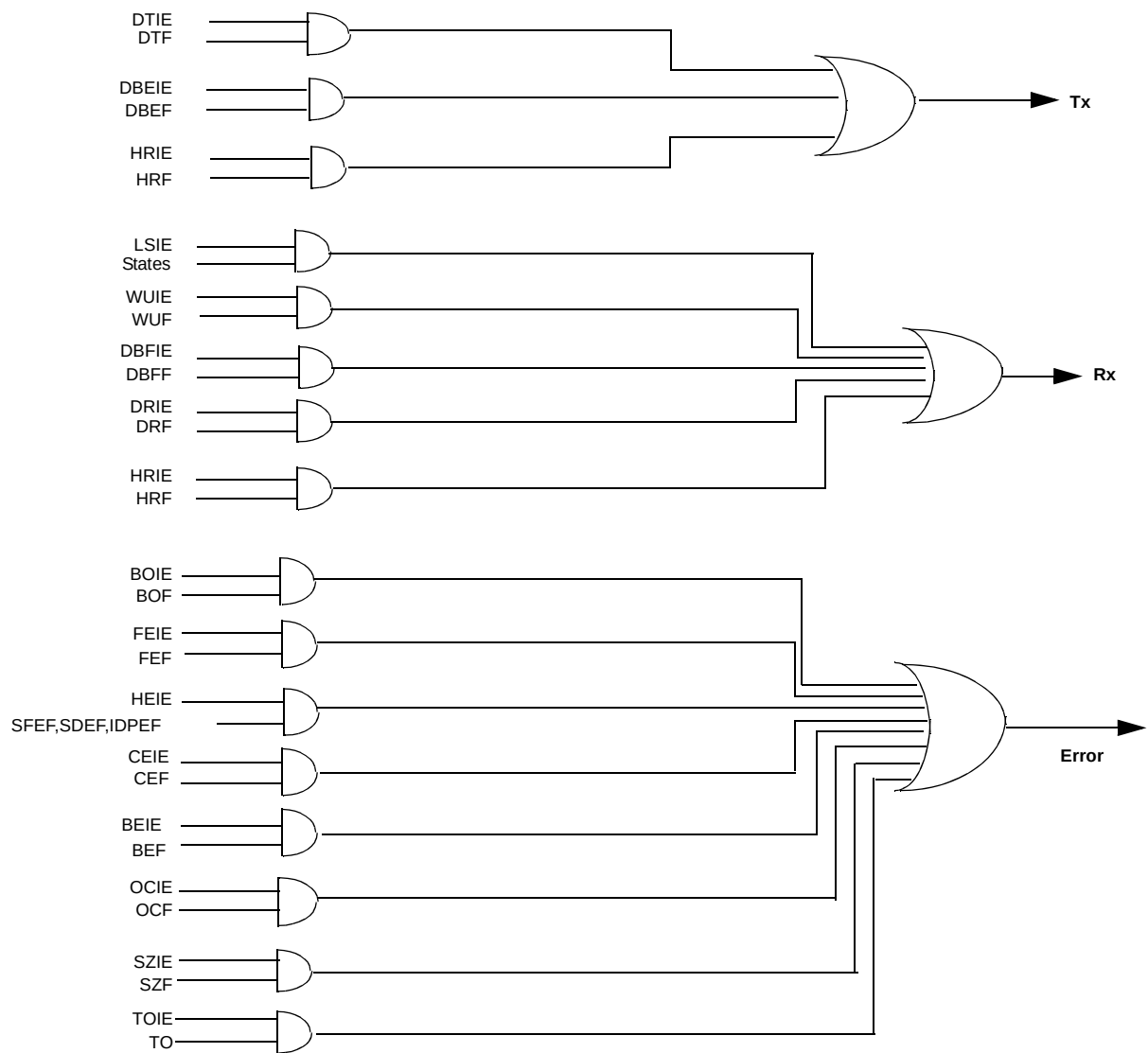
Table 1291. Interrupts

Interrupt event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error

Table 1291. Interrupts (continued)

Interrupt event	Event flag	Enable control bit	Interrupt vector
Header error	SFEF or SDEF or IDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer overrun error	BOF	BOIE	Error
UART timeout error	TO	TOIE	Error
LIN state	Sync Del, Sync Field, Identifier Field, or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data buffer full	DBFF	DBFIE	Rx
Data buffer empty	DBEF	DBEIE	Tx
Data reception complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

Figure 1383. Interrupt diagram



56.6 External signal description

56.6.1 Interface signals summary

Table 1292. Interface signals

Signal name	Active level	Reset value	In/Out	Description
IPS Bus Signals				
lps_module_en	High	0	In	Chip select
lpg_clk	—	—	In	PBRIDGE _{Ex} _CLK

Table 1292. Interface signals (continued)

Signal name	Active level	Reset value	In/Out	Description
ipg_baud_clk	—	—	In	LIN_CLK
ipg_hard_async_reset_b	High	—	In	System reset
ipg_baud_reset_b	High	—	In	Baud clock reset
ips_rwb	—	0	In	IPS read/write
ips_addr	—	—	In	IPS address
ips_wdata	—	—	In	Data write bus
ips_byte_31_24	—	—	In	Byte 24 to 31 bits select
ips_byte_23_16	High	—	In	Byte 16 to 23 bits select
ips_byte_15_8	—	—	In	Byte 8 to 15 bits select
ips_byte_7_0	High	—	In	Byte 0 to 7 bits select
ipg_stop	High	—	In	After the ongoing transaction is completed, the transmitter/receiver stay in idle mode till ipg_stop is high.
ipg_stop_ack	High	—	Out	Acknowledgement signal sent when transmitter/receiver go to idle mode when ipg_stop is asserted.
ips_rdata	—	—	Out	Data read bus
ips_xfr_err	High	—	Out	Error on accessing unused register address space and on writing read-only or reading write-only registers
DMA Tx signals				
ipd_req_tx[2**tx_ch_num-1:0]	High	—	Out	DMA Tx request
dma_ipd_complete_tx[2**tx_ch_num-1:0]	High	—	In	DMA Tx major loop done indicator
dma_done_lw_tx[2**tx_ch_num-1:0]	High	—	In	DMA Tx minor loop, last write address phase
dma_enb_req_tx[2**tx_ch_num-1:0]	High	—	In	DMA Tx enable request (content of the DMAERQ register)
DMA Rx Signals				
ipd_req_rx[2**rx_ch_num-1:0]	High	—	Out	DMA Rx request
dma_ipd_complete_rx[2**rx_ch_num-1:0]	High	—	In	DMA Rx major loop done indicator
dma_done_lw_rx[2**rx_ch_num-1:0]	High	—	In	DMA Rx minor loop, last write address phase
dma_enb_req_rx[2**rx_ch_num-1:0]	High	—	In	DMA Rx enable request (content of the DMAERQ register)
LIN signals				
ipp_ind_linrx	—	—	In	Receiver input for LIN

Table 1292. Interface signals (continued)

Signal name	Active level	Reset value	In/Out	Description
ipp_do_lintx	—	—	Out	Transmitter output for LIN
ipp_port_en_lintx	High	0	Out	Alternate function enable for TX
ipi_int_rx	High	0	Out	Receiver interrupt request
ipi_int_tx	High	0	Out	Transmitter interrupt request
ipi_int_err	High	0	Out	Error interrupt request
ipi_int_all	High	0	Out	All interrupt request for LIN
in_ps_clr_DRF ⁽¹⁾	High	0	In	These are added for PSI5-S UART interface. They are used to clear status bits.
in_ps_clr_DTF ¹	High	0	In	
in_ps_clr_PE[3:0] ¹	High	0	In	
in_ps_clr_FEF ¹	High	0	In	
in_ps_clr_BOF ¹	High	0	In	
in_ps_clr_TOF ¹	High	0	In	
in_ps_clr_RMB ¹	High	0	In	This is added for PSI5-S UART interface. This is used to clear status bit RMB.
in_ps_bit_remap ¹	High	0	In	This is added for PSI5-S UART interface. When this signal is set, it modifies the functionality of DTU, and PCE bits in Section 56.4.2.5, UART Mode Control Register (UARTCR) and SLEEP, and INIT bits in Section 56.4.2.1, LIN Control Register 1 (LINC1R)
in_ps_sleep ¹	High	1	In	When in_ps_bit_remap is set, the SLEEP bit in LINC1R is set/cleared by this signal only.
in_ps_init ¹	High	0	In	When in_ps_bit_remap is set, the INIT bit in LINC1R is set/cleared by this signal only.
out_ps_DRF ⁽²⁾	High	0	Out	These are added for PSI5-S UART interface. They reflect the value of corresponding status bits in UARTSR.
out_ps_DTF ²	High	0	Out	
out_ps_PE[3:0] ²	—	0	Out	
out_ps_FEF ²	High	0	Out	
out_ps_BOF ²	High	0	Out	
out_ps_TOF ²	High	0	Out	
out_ps_RMB ²	High	0	Out	This is added for PSI5-S UART interface. This reflects the value of status bit RMB.
out_bdr32b_rx[31:0] ²	—	0	Out	Added for PSI5-S UART interface. It reflects the value of receive buffer.

Table 1292. Interface signals (continued)

Signal name	Active level	Reset value	In/Out	Description
out_wr_tdsr ²	High	0	Out	Added for PSI5-S UART interface. It indicates the data from buffer has been moved to transmit shift register.
out_ps_mode[1:0] ²	—	0	Out	Added for PSI5-S UART interface. This signal indicates which mode LINFlexD is in. 00: SLEEP mode 01: INIT mode 10: Normal mode 11: Not applicable
Scan Inputs				
scan_mode	High	0	In	Scan enable signal
Generics				
slave	—	—	—	Note: Refer to device configuration chapter to see the value of slave used in the device.
addr_width	—	14	—	Address bus width
no_of_filters	—	—	—	Note: Refer to device configuration chapter to see the number of filters used in the device.
rx_ch_num	—	—	—	Note: Refer to device configuration chapter to see the number of rx_ch_num used in the device.
tx_ch_num	—	—	—	Note: Refer to device configuration chapter to see the number of tx_ch_num used in the device.
auto_sync	—	—	—	Note: Refer to device configuration chapter to see the value of auto_sync used in the device.

1. These inputs should be tied to '0' if PSI5-S interface is not used.
2. These outputs can be left unconnected if PSI5-S interface is not used.

- The total number of DMA channels (Tx + Rx) that can be used in SLAVE mode must be less than or equal to the number of ID filters.
- 2 DMA channels (Tx + Rx) are always required to support the DMA capability in LIN master/slave or UART mode.
- Number of Filters is independent of filtered IDs. Each filter supports more than just one ID. Wild cards are possible.
- To use only one DMA channel for Tx and one for Rx in LINFlexD mode, only two filters (one for Tx and one for Rx) have to be used in software. Wild cards are possible for more than one ID.

57 Reset Generation Module (MC_RGM)

57.1 Introduction

57.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

[Figure 1384](#) shows the MC_RGM block diagram.

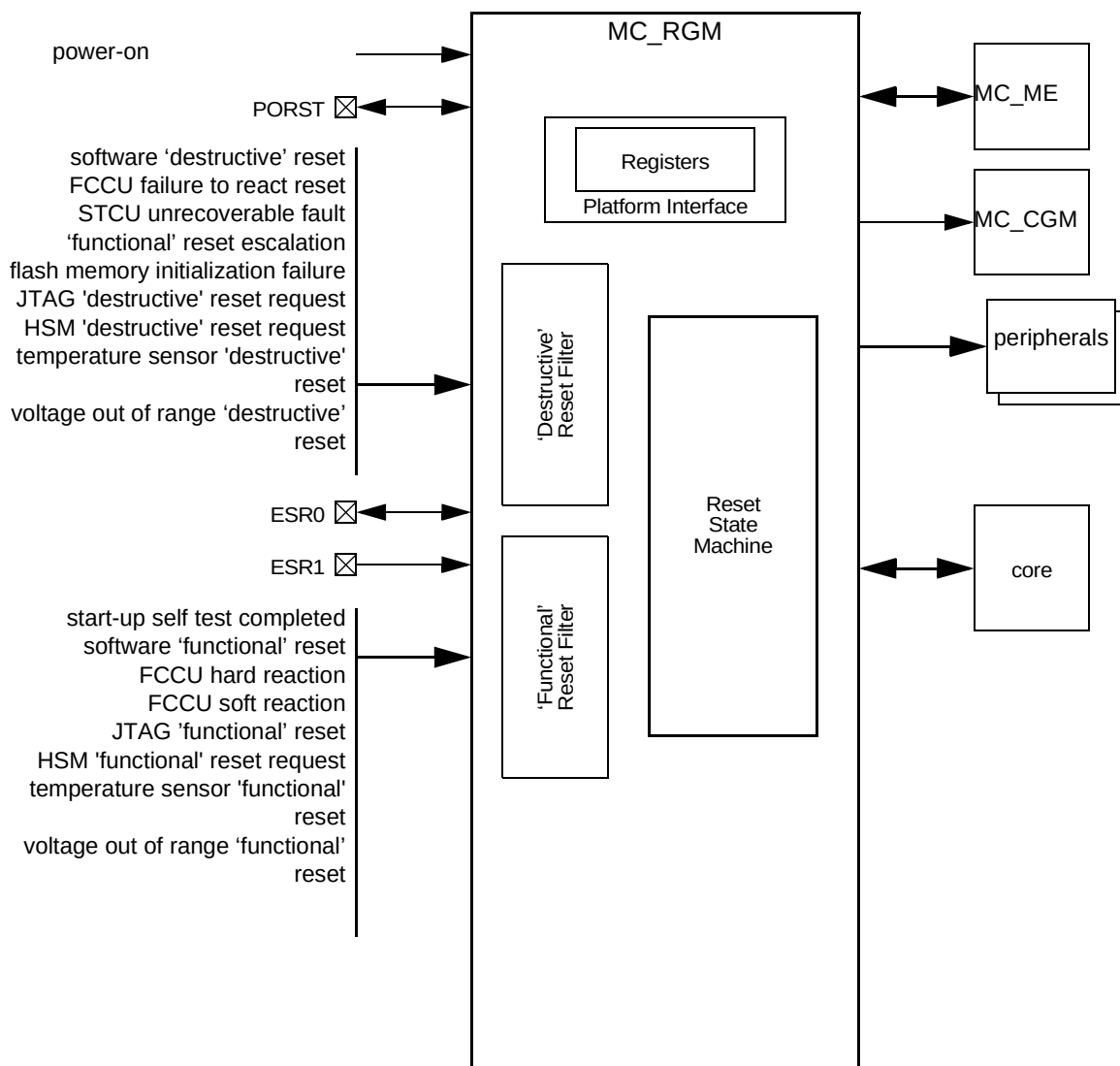


Figure 1384. MC_RGM block diagram

57.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'destructive' resets management
- 'functional' resets management
- signalling of reset events after each reset sequence (reset status flags)
- conversion of reset events to SAFE mode or interrupt request events
- short reset sequence configuration
- bidirectional reset behavior configuration
- configurable escalation of recurring 'functional' resets to 'destructive' reset
- configurable escalation of recurring 'destructive' resets to keep chip in reset state until next power-on reset

57.1.3 Reset sources

The different reset sources are organized into two families: 'destructive' and 'functional'.

- A 'destructive' reset source is associated with an event related to a critical — usually hardware — error or dysfunction. When a 'destructive' reset event occurs, the full reset sequence is applied to the chip starting from PHASE0. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be unknown except in the software 'destructive' reset and external power-on reset cases, which do preserve the system memory content if the chip is not configured to execute a self-test on power-on, 'destructive', and external resets. 'Destructive' resets are:
 - power-on reset
 - external power-on reset
 - software 'destructive' reset
 - FCCU failure to react reset
 - STCU2 unrecoverable fault
 - 'functional' reset escalation
 - flash memory initialization failure
 - JTAG 'destructive' reset request
 - HSM 'destructive' reset request
 - temperature sensor 'destructive' reset
 - voltage out of range 'destructive' reset
- A 'functional' reset source is associated with an event related to a less-critical — usually non-hardware — error or dysfunction. When a 'functional' reset event occurs, a partial reset sequence is applied to the chip starting from PHASE1. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules (e.g., debug modules, flash modules) as well as the system memory content is preserved, except on an external reset in the case where the chip is configured to execute a self-test on power-on, 'destructive', and external resets. 'Functional' resets are:
 - ESR0 external reset
 - ESR1 external reset
 - start-up self test completed

- software 'functional' reset
- FCCU hard reaction
- FCCU soft reaction
- JTAG 'functional' reset
- HSM 'functional' reset request
- temperature sensor 'functional' reset
- voltage out of range 'functional' reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e., PHASEn states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the IDLE phase. During this entire process, the MC_ME state machine is held in RESET mode. Only at the end of the reset sequence, when the IDLE phase is reached, does the MC_ME enter the DRUN mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt issued to the core (see [Section 57.3.1.2, 'Destructive' Event Reset Disable Register \(RGM_DERD\)](#) and [Section 57.3.1.3, 'Destructive' Event Alternate Request Register \(RGM_DEAR\)](#) for 'destructive' resets and [Section 57.3.1.6, 'Functional' Event Reset Disable Register \(RGM_FERD\)](#) and [Section 57.3.1.7, 'Functional' Event Alternate Request Register \(RGM_FEAR\)](#) for 'functional' resets).

57.2 External signal description

The MC_RGM interfaces to the bidirectional reset pin ESR0.

57.3 Memory map and register definition

Table 1293. MC_RGM register description

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x000	RGM_DES	'Destructive' Event Status	word	read	read/write ⁽¹⁾	read/write ¹	on page 2514
0x010	RGM_DERD	'Destructive' Event Reset Disable	word	read	read/write ⁽²⁾	read/write ⁽²⁾	on page 2516
0x020	RGM_DEAR	'Destructive' Event Alternate Request	word	read	read/write	read	on page 2518
0x030	RGM_DBRE	'Destructive' Bidirectional Reset Enable	word	read	read	read	on page 2519
0x300	RGM_FES	'Functional' Event Status	word	read	read/write ⁽¹⁾	read/write ⁽¹⁾	on page 2520
0x310	RGM_FERD	'Functional' Event Reset Disable	word	read	read/write ⁽²⁾	read/write ⁽²⁾	on page 2522

Table 1293. MC_RGM register description (continued)

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x320	RGM_FEAR	'Functional' Event Alternate Request	word	read	read/write	read/write	on page 2524
0x330	RGM_FBRE	'Functional' Bidirectional Reset Enable	word	read	read/write	read/write	on page 2525
0x340	RGM_FESS	'Functional' Event Short Sequence	word	read	read/write	read/write	on page 2526
0x604	RGM_FRET	'Functional' Reset Threshold	byte	read	read/write	read/write	on page 2528
0x608	RGM_DRET	'Destructive' Reset Escalation Threshold Register	byte	read	read/write	read/write	on page 2529
0x60C	RGM_EROEC	External Reset Output Extension Control Register	byte	read	read/write	read/write	on page 2529
0x610	RGM_PRST0	Peripheral Reset 0	word	read	read/write	read/write	on page 2530
0x614	RGM_PRST1	Peripheral Reset 1	word	read	read/write	read/write	on page 2531
0x618	RGM_PRST2	Peripheral Reset 2	word	read	read/write	read/write	on page 2531
0x61C	RGM_PRST3	Peripheral Reset 3	word	read	read/write	read/write	on page 2532
0x620	RGM_PRST4	Peripheral Reset 4	word	read	read/write	read/write	on page 2533
0x624	RGM_PRST5	Peripheral Reset 5	word	read	read/write	read/write	on page 2534
0x628	RGM_PRST6	Peripheral Reset 6	word	read	read/write	read/write	on page 2534
0x62C	RGM_PRST7	Peripheral Reset 7	word	read	read/write	read/write	on page 2535
0x630	RGM_PSTAT0	Peripheral Reset Status 0	word	read	read	read	on page 2535
0x634	RGM_PSTAT1	Peripheral Reset Status 1	word	read	read	read	on page 2536
0x638	RGM_PSTAT2	Peripheral Reset Status 2	word	read	read	read	on page 2537
0x63C	RGM_PSTAT3	Peripheral Reset Status 3	word	read	read	read	on page 2538
0x640	RGM_PSTAT4	Peripheral Reset Status 4	word	read	read	read	on page 2539

Table 1293. MC_RGM register description (continued)

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x644	RGM_PSTAT5	Peripheral Reset Status 5	word	read	read	read	on page 2540
0x648	RGM_PSTAT6	Peripheral Reset Status 6	word	read	read	read	on page 2540
0x64C	RGM_PSTAT7	Peripheral Reset Status 7	word	read	read	read	on page 2541

- individual bits cleared on writing '1'
- write once: '0' = enable, '1' = disable.

Note: Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Table 1294. MC_RGM memory map

Address	Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x000	RGM_DES	R	0	0	0	0	0	0	0	F_VOR_DEST	F_TSR_DEST	0	0	0	0	0	0
		W							w1c	w1c							
		R	F_HSM_DEST	0	0	0	0	F_JTAG_DEST	F_FIF	F_EDR	0	0	F_SUF	F_FFRR	F_SOFT_DEST	0	F_PORST
		W	w1c					w1c	w1c	w1c			w1c	w1c	w1c		w1c
0x004 ... 0x00C	reserved																

Table 1294. MC_RGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x010	RGM_DERD	R	0	0	0	0	0	0	0	D_VOR_DEST	D_TSR_DEST	0	0	0	0	0	0	0
		W																
		R	D_HSM_DEST	0	0	0	0	D_JTAG_DEST	D_FIF	D_EDR	0	0	D_SUF	D_FFRR	D_SOFT_DEST	0	D_PORST	D_POR
		W																
0x014 ... 0x01C	reserved																	
0x020	RGM_DEAR	R	0	0	0	0	0	0	0	0	AR_TSR_DEST	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AR_PORST	0
		W																
0x024 ... 0x02C	reserved																	
0x030	RGM_DBRE	R	0	0	0	0	0	0	0	BE_VOR_DEST	BE_TSR_DEST	0	0	0	0	0	0	0
		W																
		R	BE_HSM_DEST	0	0	0	0	BE_JTAG_DEST	BE_FIF	BE_EDR	0	0	BE_SUF	BE_FFRR	BE_SOFT_DEST	0	BE_PORST	BE_POR
		W																

Table 1294. MC_RGM memory map (continued)

Address	Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x034 ... 0x2FC	reserved																
0x300	RGM_FES	R	0	0	0	0	0	0	0	F_VOR_FUNC	F_TSR_FUNC	0	0	0	0	0	0
		W								w1c	w1c						
		R	F_HSM_FUNC	0	0	0	0	F_JTAG_FUNC	0	0	0	F_FCCU_SOFT	F_FCCU_HARD	0	F_SOFT_FUNC	F_ST_DONE	F_ESR1
		W	w1c					w1c				w1c	w1c		w1c	w1c	w1c
0x304 ... 0x30C	reserved																
0x310	RGM_FERD	R	0	0	0	0	0	0	0	D_VOR_FUNC	D_TSR_FUNC	0	0	0	0	0	0
		W								D_VOR_FUNC	D_TSR_FUNC						
		R	D_HSM_FUNC	0	0	0	0	D_JTAG_FUNC	0	0	0	D_FCCU_SOFT	D_FCCU_HARD	0	D_SOFT_FUNC	D_ST_DONE	D_ESR1
		W													D_SOFT_FUNC		0
0x314 ... 0x31C	reserved																

Table 1294. MC_RGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x320	RGM_FEAR	R	0	0	0	0	0	0	0	AR_VOR_FUNC	AR_TSR_FUNC	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	AR_SOFT_FUNC	0	AR_ESR1	0
		W																
		reserved																
0x324... 0x32C																		
0x330	RGM_FBRE	R	0	0	0	0	0	0	0	BE_VOR_FUNC	BE_TSR_FUNC	0	0	0	0	0	0	0
		W																
		R	BE_HSM_FUNC	0	0	0	0	BE_JTAG_FUNC	0	0	0	BE_FCCU_SOFT	BE_FCCU_HARD	0	BE_SOFT_FUNC	BE_ST_DONE	BE_ESR1	BE_ESR0
		W																
		reserved																
0x334... 0x33C																		
0x340	RGM_FESS	R	0	0	0	0	0	0	0	SS_VOR_FUNC	SS_TSR_FUNC	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	SS_JTAG_FUNC	0	0	0	SS_FCCU_SOFT	SS_FCCU_HARD	0	SS_SOFT_FUNC	SS_ST_DONE	SS_ESR1	SS_ESR0
		W																
		reserved																
0x344... 0x600																		

Table 1294. MC_RGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x604	RGM_FRET	R	0	0	0	0	FRET				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0x608	RGM_DRET	R	0	0	0	0	DRET				0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0x60C	RGM_EROE C	R	0	0	0	0	0	0	ERIS	EROEC	0	0	0	0	0	0	0	0
		W								w1c								
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0x610	RGM_PRST0	R	PIT_RTC_1_RST	PIT_RTC_0_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	SIUL_RST	0	0	0	SIPL_0_RST	0	0	0	0	0	0	0	0	0	0	0
		W																
0x614	RGM_PRST1	R	0	0	0	ADCSD_0_RST	ADCSD_2_RST	ADCSD_4_RST	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	CRC_0_RST	0	DMAMUX_0_RST	0	0	0	0
		W																

Table 1294. MC_RGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x618	RGM_PRST2	R	0	0	DSPI_12_RST	LINFlexD_0_RST	LINFlexD_1_RST	0	0	0	0	0	LINFlexD_14_RST	0	0	0	0	0
		W																
		R	0	0	0	0	0	CAN_RAM_CTR_RST	0	TTCAN_RST	0	MCAN_1_RST	MCAN_2_RST	MCAN_3_RST	0	0	0	0
		W																
0x61C	RGM_PRST3	R	ADCSAR_0_RST	0	0	0	ADCSAR_4_RST	0	0	0	0	0	0	0	0	0	0	ADCSAR_b_RST
		W																
		R	PSI5_0_RST	0	0	0	FLEXRAY_0_RST	0	0	SENT_0_RST	0	0	IIC_0_RST	0	DSPI_0_RST	DSPI_1_RST	DSPI_4_RST	0
		W																
0x620	RGM_PRST4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GTMINT_RST
		W																
0x624	RGM_PRST5	R	0	0	0	ADCSD_1_RST	ADCSD_3_RST	ADCSD_5_RST	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	FCCU_RST	0	0	CRC_1_RST	0	0	0	0	0	0
		W																

Table 1294. MC_RGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x628	RGM_PRST6	R	0	0	0	LINFlexD_2_RST	0	0	0	0	0	0	LINFlexD_15_RST	0	0	0	0	0
		W																
		R	0	0	0		0	0	0	0	0	0		0	0	0	0	
		W																
0x62C	RGM_PRST7	R	0	ADCSAR_1_RST	ADCSAR_2_RST	ADCSAR_3_RST	0	0	ADCSAR_6_RST	ADCSAR_7_RST	0	0	0	0	0	0	0	0
		W																
		R	PSI5_1_RST	0	0	0	0	0	0	SENT_1_RST	0	0	0	0	DSPI_2_RST	DSPI_3_RST	DSPI_5_RST	0
		W																
0x630	RGM_PSTAT0	R	PIT_RTC_1_STAT	PIT_RTC_0_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	SIUL_STAT	0	0	0	SIPI_0_STAT	0	LFAST_0_STAT	0	0	0	0	0	0	0	0	
		W																

Table 1294. MC_RGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x634	RGM_PSTAT 1	R	0	0	0	ADCSD_0_STAT	ADCSD_2_STAT	ADCSD_4_STAT	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	CRC_0_STAT	0	DMAMUX_0_STAT	0	0	0	0
		W																
0x638	RGM_PSTAT 2	R	0	0	DSPI_12_STAT	LINFlexD_0_STAT	LINFlexD_1_STAT	0	0	0	0	0	LINFlexD_14_STAT	0	0	0	0	0
		W																
		R	0	0	0	0	0	CAN_RAM_CTR_STAT	0	TTCAN_STAT	0	MCAN_1_STAT	MCAN_2_STAT	MCAN_3_STAT	0	0	0	0
		W																

Table 1294. MC_RGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x63C	RGM_PSTAT 3	R	ADCSAR_0_STAT	0	0	0	ADCSAR_4_STAT	0	0	0	0	0	0	0	0	0	0	ADCSAR_b_STAT
		W																
		R	PSI5_0_STAT	0	0	0	FLEXRAY_0_STAT	0	0	SENT_0_STAT	0	0	IIC_0_STAT	0	DSPI_0_STAT	DSPI_1_STAT	DSPI_4_STAT	0
		W																
0x640	RGM_PSTAT 4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GTMINT_STAT
		W																
0x644	RGM_PSTAT 5	R	0	0	0	ADCSD_1_STAT	ADCSD_3_STAT	ADCSD_5_STAT	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	FCCU_STAT	0	0	CRC_1_STAT	0	0	0	0	0	0
		W																

Table 1294. MC_RGM memory map (continued)

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x648	RGM_PSTAT 6	R	0	0	0	LINFlexD_2_STAT	0	0	0	0	0	0	LINFlexD_15_STAT	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
0x64C	RGM_PSTAT 7	R	0	ADCSAR_1_STAT	ADCSAR_2_STAT	ADCSAR_3_STAT	0	0	ADCSAR_6_STAT	ADCSAR_7_STAT	0	0	0	0	0	0	0	0
		W																
		R	PSI5_1_STAT	0	0	0	0	0	0	SENT_1_STAT	0	0	0	0	DSPI_2_STAT	DSPI_3_STAT	DSPI_5_STAT	0
		W																

57.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the RGM_DES[8:15] register bits may be accessed as a word at address offset 0x000, as a half-word at address offset 0x002, or as a byte at address offset 0x003.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

57.3.1.1 'Destructive' Event Status Register (RGM_DES)

This register contains the status of the 'destructive' reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write '1'.

This register is reset only on power-on.

Figure 1385. 'Destructive' Event Status Register (RGM_DES)

Address 0x000

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	F_VOR_DEST	F_TSR_DEST	0	0	0	0	0	0	0
W								w1c	w1c							
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_HSM_DEST	0	0	0	0	F_JTAG_DEST	F_FIF	F_EDR	0	0	F_SUF	F_FFRR	F_SOFT_DEST	0	F_PORST	F_POR
W	w1c					w1c	w1c	w1c			w1c	w1c	w1c		w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 1295. RGM_DES field descriptions

Field	Description
F_VOR_DEST	Flag for voltage out of range 'destructive' reset 0 No voltage out of range 'destructive' reset event has occurred since the last clear 1 A voltage out of range 'destructive' reset event has occurred
F_TSR_DEST	Flag for temperature sensor 'destructive' reset 0 No temperature sensor 'destructive' reset event has occurred since the last clear 1 A temperature sensor 'destructive' reset event has occurred
F_HSM_DEST	Flag for HSM 'destructive' reset request 0 No HSM 'destructive' reset request event has occurred since the last clear 1 A HSM 'destructive' reset request event has occurred
F_JTAG_DEST	Flag for JTAG 'destructive' reset request 0 No JTAG 'destructive' reset request event has occurred since the last clear 1 A JTAG 'destructive' reset request event has occurred
F_FIF	Flag for flash memory initialization failure 0 No flash memory initialization failure event has occurred since the last clear 1 A flash memory initialization failure event has occurred
F_EDR	Flag for 'functional' reset escalation 0 No 'functional' reset escalation event has occurred since the last clear 1 A 'functional' reset escalation event has occurred

Table 1295. RGM_DES field descriptions (continued)

Field	Description
F_SUF	Flag for STCU2 unrecoverable fault 0 No STCU2 unrecoverable fault event has occurred since the last clear 1 A STCU2 unrecoverable fault event has occurred
F_FFRR	Flag for FCCU failure to react reset 0 No FCCU failure to react reset event has occurred since the last clear 1 A FCCU failure to react reset event has occurred
F_SOFT_DEST	Flag for software 'destructive' reset 0 No software 'destructive' reset event has occurred since the last clear 1 A software 'destructive' reset event has occurred
F_PORST	Flag for external power-on reset 0 No external power-on reset has occurred since the last clear 1 A external power-on reset event has occurred
F_POR	Flag for Power-On reset 0 No power-on event has occurred since the last clear 1 A power-on event has occurred

57.3.1.2 'Destructive' Event Reset Disable Register (RGM_DERD)

This register provides dedicated bits to disable particular 'destructive' reset sources. When a 'destructive' reset source is disabled, the associated 'destructive' event will trigger either a SAFE mode request or an interrupt request (see [Section 57.3.1.3, 'Destructive' Event Alternate Request Register \(RGM_DEAR\)](#)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after a 'destructive' or power-on reset.

This register is reset only on power-on.

Caution: It is important to clear the RGM_DES register before setting any of the bits in the RGM_DERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Figure 1386. 'Destructive' Event Reset Disable Register (RGM_DERD)

Address 0x010

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	D_VOR_DEST	D_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	D_HSM_DEST	0	0	0	0	D_JTAG_DEST	D_FIF	D_EDR	0	0	D_SUF	D_FFRR	D_SOFT_DEST	0	D_PORST	D_POR
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1296. RGM_DERD field descriptions

Field	Description
D_VOR_DEST	Disable voltage out of range 'destructive' reset 0 A voltage out of range 'destructive' reset event triggers a reset sequence
D_TSR_DEST	Disable temperature sensor 'destructive' reset 0 A temperature sensor 'destructive' reset event triggers a reset sequence 1 A temperature sensor 'destructive' reset event generates either a SAFE mode or an interrupt request depending on the values in RGM_DEAR
D_HSM_DEST	Disable HSM 'destructive' reset request 0 A HSM 'destructive' reset request event triggers a reset sequence
D_JTAG_DEST	Disable JTAG 'destructive' reset request 0 A JTAG 'destructive' reset request event triggers a reset sequence
D_FIF	Disable flash memory initialization failure 0 A flash memory initialization failure event triggers a reset sequence
D_EDR	Disable 'functional' reset escalation 0 A 'functional' reset escalation event triggers a reset sequence
D_SUF	Disable STCU2 unrecoverable fault 0 A STCU2 unrecoverable fault event triggers a reset sequence
D_FFRR	Disable FCCU failure to react reset 0 A FCCU failure to react reset event triggers a reset sequence

Table 1296. RGM_DERD field descriptions (continued)

Field	Description
D_SOFT_DEST	Disable software 'destructive' reset 0 A software 'destructive' reset triggers a reset sequence
D_PORST	Disable external power-on reset 0 An external power-on reset event triggers a reset sequence 1 An external power-on reset event generates either a SAFE mode or an interrupt request depending on the values in RGM_DEAR
D_POR	Disable Power-On reset 0 A power-on event triggers a reset sequence

57.3.1.3 'Destructive' Event Alternate Request Register (RGM_DEAR)

This register defines an alternate request to be generated when a reset on a 'destructive' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Figure 1387. 'Destructive' Event Alternate Request Register (RGM_DEAR)

Address 0x020

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	AR_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AR_PORST	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1297. RGM_DEAR field descriptions

Field	Description
AR_TSR_DEST	Alternate Request for temperature sensor 'destructive' reset 0 Generate a SAFE mode request on a temperature sensor 'destructive' reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor 'destructive' reset event if the reset is disabled
AR_PORST	Alternate Request for external power-on reset 0 Generate a SAFE mode request on an external power-on reset event if the reset is disabled 1 Generate an interrupt request on an external power-on reset event if the reset is disabled

57.3.1.4 Destructive' Bidirectional Reset Enable Register (RGM_DBRE)

This register enables the generation of an external reset on 'destructive' reset. It can be accessed in read only in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Figure 1388. Destructive' Bidirectional Reset Enable Register (RGM_DBRE)

Address 0x030

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	BE_VOR_DEST	BE_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE_HSM_DEST	0	0	0	0	BE_JTAG_DEST	BE_FIF	BE_EDR	0	0	BE_SUF	BE_FFRR	BE_SOFT_DEST	0	BE_PORST	BE_POR
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1298. RGM_DBRE field descriptions

Field	Description
BE_VOR_DEST	Bidirectional Reset Enable for voltage out of range 'destructive' reset 0ESR0_B is asserted on a voltage out of range 'destructive' reset event
BE_TSR_DEST	Bidirectional Reset Enable for temperature sensor 'destructive' reset 0ESR0_B is asserted on a temperature sensor 'destructive' reset event if the reset is enabled
BE_HSM_DEST	Bidirectional Reset Enable for HSM 'destructive' reset request 0ESR0_B is asserted on a HSM 'destructive' reset request event 1 ESR0_B is not asserted on a HSM 'destructive' reset request event
BE_JTAG_DEST	Bidirectional Reset Enable for JTAG 'destructive' reset request 0ESR0_B is asserted on a JTAG 'destructive' reset request event 1 ESR0_B is not asserted on a JTAG 'destructive' reset request event
BE_FIF	Bidirectional Reset Enable for flash memory initialization failure 0ESR0_B is asserted on a flash memory initialization failure event
BE_EDR	Bidirectional Reset Enable for 'functional' reset escalation 0ESR0_B is asserted on a 'functional' reset escalation event
BE_SUF	Bidirectional Reset Enable for STCU2 unrecoverable fault 0ESR0_B is asserted on a STCU2 unrecoverable fault event
BE_FFRR	Bidirectional Reset Enable for FCCU failure to react reset 0ESR0_B is asserted on a FCCU failure to react reset event
BE_SOFT_DEST	Bidirectional Reset Enable for software 'destructive' reset 0ESR0_B is asserted on a software 'destructive' reset event
BE_PORST	Bidirectional Reset Enable for external power-on reset 0ESR0_B is asserted on an external power-on reset event if the reset is enabled
BE_POR	Bidirectional Reset Enable for Power-On reset 0ESR0_B is asserted on a Power-On reset event

57.3.1.5 'Functional' Event Status Register (RGM_FES)

This register contains the status of the 'functional' reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write '1' if the triggering event has already been cleared at the source.

This register is reset only on power-on.

Note: *If a 'functional' reset source is configured to generate a SAFE mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles before it clears the associated RGM_FES status bit in order to avoid multiple SAFE mode requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the RGM_FES has been properly cleared, and if not, clear it again.*

Figure 1389. 'Functional' Event Status Register (RGM_FES)

Address 0x300

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	F_VOR_FUNC	F_TSR_FUNC	0	0	0	0	0	0	0
W								w1c	w1c							
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_HSM_FUNC	0	0	0	0	F_JTAG_FUNC	0	0	0	F_FCCU_SOFT	F_FCCU_HARD	0	F_SOFT_FUNC	F_ST_DONE	F_ESR1	F_ESR0
W	w1c					w1c				w1c	w1c		w1c	w1c	w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1299. RGM_FES field descriptions

Field	Description
F_VOR_FUNC	Flag for voltage out of range 'functional' reset 0 No voltage out of range 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A voltage out of range 'functional' reset event has occurred
F_TSR_FUNC	Flag for temperature sensor 'functional' reset 0 No temperature sensor 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A temperature sensor 'functional' reset event has occurred
F_HSM_FUNC	Flag for HSM 'functional' reset request 0 No HSM 'functional' reset request event has occurred since either the last clear or the last power-on reset assertion 1 A HSM 'functional' reset request event has occurred
F_JTAG_FUNC	Flag for JTAG 'functional' reset 0 No JTAG 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A JTAG 'functional' reset event has occurred
F_FCCU_SOFT	Flag for FCCU soft reaction 0 No FCCU soft reaction event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU soft reaction event has occurred

Table 1299. RGM_FES field descriptions (continued)

Field	Description
F_FCCU_HARD	Flag for FCCU hard reaction reset 0 No FCCU hard reaction reset event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU hard reaction reset event has occurred
F_SOFT_FUNC	Flag for software 'functional' reset 0 No software 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A software 'functional' reset event has occurred
F_ST_DONE	Flag for self test completed 0 No self test completed event has occurred since either the last clear or the last power-on reset assertion 1 A self test completed event has occurred
F_ESR1	Flag for ESR1 External Reset 0 No ESR1 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR1 external reset event has occurred
F_ESR0	Flag for ESR0 External Reset 0 No ESR0 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR0 external reset event has occurred

57.3.1.6 'Functional' Event Reset Disable Register (RGM_FERD)

This register provides dedicated bits to disable 'functional' reset sources. When a 'functional' reset source is disabled, the associated 'functional' event will trigger either a SAFE mode request or an interrupt request (see [Section 57.3.1.7, 'Functional' Event Alternate Request Register \(RGM_FEAR\)](#)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

This register is reset only on power-on and any 'destructive' reset.

Caution: It is important to clear the RGM_FES register before setting any of the bits in the RGM_FERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Figure 1390. 'Functional' Event Reset Disable Register (RGM_FERD)

Address 0x310

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	D_VOR_FUNC	D_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	D_HSM_FUNC	0	0	0	0	D_JTAG_FUNC	0	0	0	D_FCCU_SOFT	D_FCCU_HARD	0	D_SOFT_FUNC	D_ST_DONE	D_ESR1	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1300. RGM_FERD field descriptions

Field	Description
D_VOR_FUNC	Disable voltage out of range 'functional' reset 0 A voltage out of range 'functional' reset event triggers a reset sequence 1 A voltage out of range 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC
D_TSR_FUNC	Disable temperature sensor 'functional' reset 0 A temperature sensor 'functional' reset event triggers a reset sequence 1 A temperature sensor 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_JTAG_FUNC
D_HSM_FUNC	Disable HSM 'functional' reset request 0 A HSM 'functional' reset request event triggers a reset sequence
D_JTAG_FUNC	Disable JTAG 'functional' reset 0 A JTAG 'functional' reset event triggers a reset sequence
D_FCCU_SOFT	Disable FCCU soft reaction 0 A FCCU soft reaction event triggers a reset sequence
D_FCCU_HARD	Disable FCCU hard reaction reset 0 A FCCU hard reaction reset event triggers a reset sequence
D_SOFT_FUNC	Disable software 'functional' reset 0 A software 'functional' reset event triggers a reset sequence

Table 1300. RGM_FERD field descriptions (continued)

Field	Description
D_ST_DONE	Disable self test completed 0 A self test completed event triggers a reset sequence
D_ESR1	Disable ESR1 External Reset 0 An ESR1 external reset event triggers a reset sequence 1 An ESR1 external reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_ESR1

57.3.1.7 'Functional' Event Alternate Request Register (RGM_FEAR)

This register defines an alternate request to be generated when a reset on a 'functional' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on and any 'destructive' reset.

Figure 1391. 'Functional' Event Alternate Request Register (RGM_FEAR)

Address 0x320

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	AR_VOR_FUNC	AR_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AR_ESR1	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1301. RGM_FEAR field descriptions

Field	Description
AR_VOR_FUNC	Alternate Request for voltage out of range 'functional' reset 0 Generate a SAFE mode request on a voltage out of range 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a voltage out of range 'functional' reset event if the reset is disabled
AR_TSR_FUNC	Alternate Request for temperature sensor 'functional' reset 0 Generate a SAFE mode request on a temperature sensor 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor 'functional' reset event if the reset is disabled
AR_ESR1	Alternate Request for ESR1 External Reset 0 Generate a SAFE mode request on an ESR1 external reset event if the reset is disabled 1 Generate an interrupt request on an ESR1 external reset event if the reset is disabled

57.3.1.8 'Functional' Bidirectional Reset Enable Register (RGM_FBRE)

This register enables the generation of an external reset on 'functional' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Figure 1392. 'Functional' Bidirectional Reset Enable Register (RGM_FBRE)

Address 0x330

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	BE_VOR_FUNC	BE_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE_HSM_FUNC	0	0	0	0	BE_JTAG_FUNC	0	0	0	BE_FCCU_SOFT	BE_FCCU_HARD	0	BE_SOFT_FUNC	BE_ST_DONE	BE_ESR1	BE_ESR0
W																
'Dest' Reset	1	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0

Table 1302. RGM_FBRE field descriptions

Field	Description
BE_VOR_FUNC	Bidirectional Reset Enable for voltage out of range 'functional' reset 0 ESR0_B is asserted on a voltage out of range 'functional' reset event if the reset is enabled 1 ESR0_B is not asserted on a voltage out of range 'functional' reset event
BE_TSR_FUNC	Bidirectional Reset Enable for temperature sensor 'functional' reset 0 ESR0_B is asserted on a temperature sensor 'functional' reset event if the reset is enabled 1 ESR0_B is not asserted on a temperature sensor 'functional' reset event
BE_HSM_FUNC	Bidirectional Reset Enable for HSM 'functional' reset request 0 ESR0_B is asserted on a HSM 'functional' reset request event 1 ESR0_B is not asserted on a HSM 'functional' reset request event
BE_JTAG_FUNC	Bidirectional Reset Enable for JTAG 'functional' reset 0 ESR0_B is asserted on a JTAG 'functional' reset event if the reset is enabled 1 ESR0_B is not asserted on a JTAG 'functional' reset event
BE_FCCU_SOFT	Bidirectional Reset Enable for FCCU soft reaction 0 ESR0_B is asserted on a FCCU soft reaction event 1 ESR0_B is not asserted on a FCCU soft reaction event
BE_FCCU_HARD	Bidirectional Reset Enable for a FCCU hard reaction 0 ESR0_B is asserted on a FCCU hard reaction reset event 1 ESR0_B is not asserted on a FCCU hard reaction reset event
BE_SOFT_FUNC	Bidirectional Reset Enable for software 'functional' reset 0 ESR0_B is asserted on a software 'functional' reset event if the reset is enabled 1 ESR0_B is not asserted on a software 'functional' reset event
BE_ST_DONE	Bidirectional Reset Enable for self test completed 0 ESR0_B is asserted on a self test completed event
BE_ESR1	Bidirectional Reset Enable for ESR1 External Reset 0 ESR0_B is asserted on an ESR1 external reset event if the reset is enabled 1 ESR0_B is not asserted on an ESR1 external reset event
BE_ESR0	Bidirectional Reset Enable for ESR0 External Reset 0 ESR0_B is asserted on an ESR0 external reset event if the reset is enabled

57.3.1.9 'Functional' Event Short Sequence Register (RGM_FESS)

This register defines which reset sequence will be done when a 'functional' reset sequence is triggered. The 'functional' reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2.

Note: This could be useful for fast reset sequence, for example to skip flash memory reset.

It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Figure 1393. 'Functional' Event Short Sequence Register (RGM_FESS)

Address 0x340

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	SS_VOR_FUNC	SS_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SS_HSM_FUNC	0	0	0	0	SS_JTAG_FUNC	0	0	0	SS_FCCU_SOFT	SS_FCCU_HARD	0	SS_SOFT_FUNC	SS_ST_DONE	SS_ESR1	SS_ESR0
W																
'Dest' Reset	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 1303. RGM_FESS field descriptions

Field	Description
SS_VOR_FUNC	Short Sequence for voltage out of range 'functional' reset 0 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
SS_TSR_FUNC	Short Sequence for temperature sensor 'functional' reset 0 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
SS_HSM_FUNC	Short Sequence for HSM 'functional' reset request 1 The reset sequence triggered by a HSM 'functional' reset request event will start from PHASE3, skipping PHASE1 and PHASE2
SS_JTAG_FUNC	Short Sequence for JTAG 'functional' reset 0 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
SS_FCCU_SOFT	Short Sequence for FCCU soft reaction 1 The reset sequence triggered by a FCCU soft reaction event will start from PHASE3, skipping PHASE1 and PHASE2

Table 1303. RGM_FESS field descriptions (continued)

Field	Description
SS_FCCU_HARD	Short Sequence for a FCCU hard reaction 0 The reset sequence triggered by a FCCU hard reaction reset event will start from PHASE1
SS_SOFT_FUNC	Short Sequence for software 'functional' reset 0 The reset sequence triggered by a software 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a software 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
SS_ST_DONE	Short Sequence for self test completed 0 The reset sequence triggered by a self test completed event will start from PHASE1
SS_ESR1	Short Sequence for ESR1 External Reset 0 The reset sequence triggered by an ESR1 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR1 external reset event if the reset is enabled will start from PHASE3, skipping PHASE1 and PHASE2
SS_ESR0	Short Sequence for ESR0 External Reset 0 The reset sequence triggered by an ESR0 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR0 external reset event if the reset is enabled will start from PHASE3, skipping PHASE1 and PHASE2

57.3.1.10 'Functional' Reset Escalation Threshold Register (RGM_FRET)

This register sets the threshold for 'functional' reset escalation to a 'destructive' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the FRET field will enable the 'functional' reset escalation function. Writing any value to this register will reset the 'functional' reset counter. See [Section 57.4.6, 'Functional' reset escalation](#) for details on the 'functional' reset escalation function.

This register is reset only on power-on and any 'destructive' reset.

Figure 1394. 'Functional' Reset Escalation Threshold Register (RGM_FRET)

Address 0x604				Access: User read, Supervisor read/write, Test read/write				
	0	1	2	3	4	5	6	7
R	0	0	0	0	FRET			
W								
'Dest' Reset	0	0	0	0	0	1	0	0

Table 1304. 'Functional' Reset Escalation Threshold Register (RGM_FRET) field descriptions

Field	Description
FRET	'Functional' Reset Escalation Threshold — If the value of this field is 0, the 'functional' reset escalation function is disabled. Any other value is the number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand.

57.3.1.11 ‘Destructive’ Reset Escalation Threshold Register (RGM_DRET)

This register sets the threshold for ‘destructive’ reset escalation keeping the chip in the reset state until the next power-on reset triggers a new reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the DRET field will enable the ‘destructive’ reset software escalation function. Writing any value to this register will reset the ‘destructive’ reset counter. See [Section 57.4.7, ‘Destructive’ reset escalation](#), for details on the ‘destructive’ reset escalation function.

This register is reset only on power-on.

Figure 1395. ‘Destructive’ Reset Escalation Threshold Register (RGM_DRET)

Address 0x608				Access: User read, Supervisor read/write, Test read/write				
	0	1	2	3	4	5	6	7
R	0	0	0	0	DRET			
W								
POR	0	0	0	0	1	0	0	0

Table 1305. RGM_DRET field descriptions

Field	Description
DRET	‘Destructive’ Reset Escalation Threshold — If the value of this field is 0, the ‘destructive’ reset escalation function is disabled. Any other value is the number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand.

57.3.1.12 External Reset Output Extension Control Register (RGM_EROEC)

This register controls the assertion of the ESR0_B pin after the reset sequence has completed. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. The register bit is cleared on write ‘1’.

Clearing the EROEC bit will cause the MC_RGM to stop asserting the ESR0_B pin within two 16 MHz internal RC oscillator clock cycles. The ESR0_B pin cannot be asserted by writing to this register.

This register also provides the current value sampled on ESR0 via the ERIS bit. This is useful for software to poll if, for example, ESR0_B is used in the system to synchronize the application software with an external event.

This register is reset only on power-on.

Figure 1396. External Reset Output Extension Control Register (RGM_EROEC)

Address 0x60C

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	ERIS	EROEC
W								w1c
POR	0	0	0	0	0	0	0	1

Table 1306. RGM_EROEC field descriptions

Field	Description
ERIS	External Reset Input Status 0 A '0' is being observed on ESR0 1 A '1' is being observed on ESR0
EROEC	External Reset Output Assertion Control 0 ESR0_B is not being asserted by the MC_RGM 1 ESR0_B is being asserted by the MC_RGM

57.3.1.13 Peripheral Reset Register 0 (RGM_PRST0)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See [Table 1307](#) for details.

Figure 1397. Peripheral Reset Register 0 (RGM_PRST0)

Address 0x610

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PIT_RTC_1_RST	PIT_RTC_0_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIUL_RST	0	0	0	SIPI_0_RST	0	LFAST_0_RST	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.14 Peripheral Reset Register 1 (RGM_PRST1)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See [Table 1307](#) for details.

Figure 1398. Peripheral Reset Register 1 (RGM_PRST1)

Address 0x614				Access: User read, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	0	0	ADCSD_0_RST	ADCSD_2_RST	ADCSD_4_RST	0	0	0	0	0	0	0	0	0	0			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CRC_0_RST	0	DMAMUX_0_RST	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.15 Peripheral Reset Register 2 (RGM_PRST2)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See [Table 1307](#) for details.

In addition to resetting the CAN_RAM_CTR module, writing a '1' to the CAN_RAM_CTR_RST bit will also reset the M_CAN_1, M_CAN_2, and M_CAN_3 modules and keep them in reset until a '0' is written to this bit. M_CAN_1, M_CAN_2, and M_CAN_3 can still be reset individually via their individual bits.

Figure 1399. Peripheral Reset Register 2 (RGM_PRST2)

Address 0x618Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	DSPI_12_RST	LINFlexD_0_RST	LINFlexD_1_RST	0	0	0	0	0	LINFlexD_14_RST	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_CTR_RST	0	TTCAN_RST	0	MCAN_1_RST	MCAN_2_RST	MCAN_3_RST	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.16 Peripheral Reset Register 3 (RGM_PRST3)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See [Table 1307](#) for details.



Figure 1400. Peripheral Reset Register 3 (RGM_PRST3)

Address 0x61C

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADCSAR_0_RST	0	0	0	ADCSAR_4_RST	0	0	0	0	0	0	0	0	0	0	ADCSAR_b_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_0_RST	0	0	0	FLEXRAY_0_RST	0	0	SENT_0_RST	0	0	IIC_0_RST	0	DSPI_0_RST	DSPI_1_RST	DSPI_4_RST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.17 Peripheral Reset Register 4 (RGM_PRST4)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See [Table 1307](#) for details.

Figure 1401. Peripheral Reset Register 4 (RGM_PRST4)

Address 0x620

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GTMINT_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.18 Peripheral Reset Register 5 (RGM_PRST5)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See [Table 1307](#) for details.

Figure 1402. Peripheral Reset Register 5 (RGM_PRST5)

Address 0x624				Access: User read, Supervisor read/write, Test read/write																								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15												
R	0	0	0				0	0	0	0	0	0	0	0	0	0												
W				ADCSD_1_RST	ADCSD_3_RST	ADCSD_5_RST																						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31												
R	0	0	0	0	0	0		0	0		0	0	0	0	0	0												
W							FCCU_RST			CRC_1_RST																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

57.3.1.19 Peripheral Reset Register 6 (RGM_PRST6)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See [Table 1307](#) for details.

Figure 1403. Peripheral Reset Register 6 (RGM_PRST6)

Address 0x628				Access: User read, Supervisor read/write, Test read/write																								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15												
R	0	0	0		0	0	0	0	0	0		0	0	0	0	0												
W				LINFlexD_2_RST							LINFlexD_15_RST																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31												
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
W																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

57.3.1.20 Peripheral Reset Register 7 (RGM_PRST7)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See [Table 1307](#) for details.

Figure 1404. Peripheral Reset Register 7 (RGM_PRST7)

Address 0x62C				Access: User read, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	ADCSAR_1_RST	ADCSAR_2_RST	ADCSAR_3_RST	0	0	ADCSAR_6_RST	ADCSAR_7_RST	0	0	0	0	0	0	0	0			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_1_RST	0	0	0	0	0	0	SENT_1_RST	0	0	0	0	DSPI_2_RST	DSPI_3_RST	DSPI_5_RST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1307. Peripheral Reset Registers (RGM_PRSTn) field descriptions

Field	Description
<periph>_RST	Peripheral reset — Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0no forced reset of peripheral 1no forced reset of peripheral

Caution: Use the RGM_PRSTn registers with care. See [Section 57.4.8, Individual peripheral resets](#) for the proper sequence to prevent unexpected chip behavior.

57.3.1.21 Peripheral Reset Register 0 (RGM_PSTAT0)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See [Table 1308](#) for details.

Figure 1405. Peripheral Reset Register 0 (RGM_PSTAT0)

Address 0x610Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PIT_RTC_1_STAT	PIT_RTC_0_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIUL_STAT	0	0	0	SIPI_0_STAT	0	LFAST_0_STAT	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.22 Peripheral Reset Register 1 (RGM_PSTAT1)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See [Table 1308](#) for details.

Figure 1406. Peripheral Reset Register 1 (RGM_PSTAT1)

Address 0x614 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	ADCSD_0_STAT	ADCSD_2_STAT	ADCSD_4_STAT	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CRC_0_STAT	0	DMAMUX_0_STAT	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.23 Peripheral Reset Register 2 (RGM_PSTAT2)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See [Table 1308](#) for details.

Figure 1407. Peripheral Reset Register 2 (RGM_PSTAT2)

Address 0x618

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	DSPI_12_STAT	LINFlexD_0_STAT	LINFlexD_1_STAT	0	0	0	0	0	LINFlexD_14_STAT	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_CTR_STAT	0	TTCAN_STAT	0	MCAN_1_STAT	MCAN_2_STAT	MCAN_3_STAT	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.24 Peripheral Reset Register 3 (RGM_PSTAT3)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See [Table 1308](#) for details.

Figure 1408. Peripheral Reset Register 3 (RGM_PSTAT3)

Address 0x61C

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADCSAR_0_STAT	0	0	0	ADCSAR_4_STAT	0	0	0	0	0	0	0	0	0	0	ADCSAR_b_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_0_STAT	0	0	0	FLEXRAY_0_STAT	0	0	SENT_0_STAT	0	0	IIC_0_STAT	0	DSPI_0_STAT	DSPI_1_STAT	DSPI_4_STAT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.25 Peripheral Reset Register 4 (RGM_PSTAT4)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See [Table 1308](#) for details.

Figure 1409. Peripheral Reset Register 4 (RGM_PSTAT4)

Address 0x620

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GTMINT_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.26 Peripheral Reset Register 5 (RGM_PSTAT5)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See [Table 1308](#) for details.

Figure 1410. Peripheral Reset Register 5 (RGM_PSTAT5)

Address 0x624				Access: User read, Supervisor read, Test read															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	0	0	ADCSD_1_STAT	ADCSD_3_STAT	ADCSD_5_STAT	0	0	0	0	0	0	0	0	0	0			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FCCU_STAT	0	0	CRC_1_STAT	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.27 Peripheral Reset Register 6 (RGM_PSTAT6)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See [Table 1308](#) for details.

Figure 1411. Peripheral Reset Register 6 (RGM_PSTAT6)

Address 0x628

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFlexD_2_STAT	0	0	0	0	0	0	LINFlexD_15_STAT	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.3.1.28 Peripheral Reset Register 7 (RGM_PSTAT7)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See [Table 1308](#) for details.

Figure 1412. Peripheral Reset Register 7 (RGM_PSTAT7)

Address 0x62C

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	ADCSAR_1_STAT	ADCSAR_2_STAT	ADCSAR_3_STAT	0	0	ADCSAR_6_STAT	ADCSAR_7_STAT	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_1_STAT	0	0	0	0	0	0	SENT_1_STAT	0	0	0	0	DSPI_2_STAT	DSPI_3_STAT	DSPI_5_STAT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1308. Peripheral Reset Registers (RGM_STATn) field descriptions

Field	Description
<periph>_STAT	Peripheral reset status. 0 peripheral is not in reset 1 peripheral is in reset

57.4 Functional description

57.4.1 Reset state machine

The main role of the MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in [Table 1309](#).

Table 1309. MC_RGM reset implications

Source	What Gets Reset	External Reset Assertion ⁽¹⁾
power-on reset	all - executes a start-up self test if enabled	yes
'destructive' resets	all except some clock/reset management - executes a start-up self test if enabled	yes
long external (ESR0) reset	all except some clock/reset management and debug - executes a start-up self test if enabled	programmable ⁽²⁾
long external (ESR0) and long 'functional' resets	all except some clock/reset management, STCU, FCCU, and debug	programmable ⁽²⁾
shortened 'functional' resets ⁽³⁾	flip-flops except some clock/reset management, STCU, FCCU, and debug	programmable ⁽²⁾

1. 'external reset assertion' means that the ESR0_B pin is asserted by the MC_RGM from the start of the reset sequence until the end of reset PHASE3
2. the assertion of the external reset is controlled via the RGM_FBRE register
3. the short sequence is enabled via the RGM_FESS register

Note: JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in [Figure 1413](#).

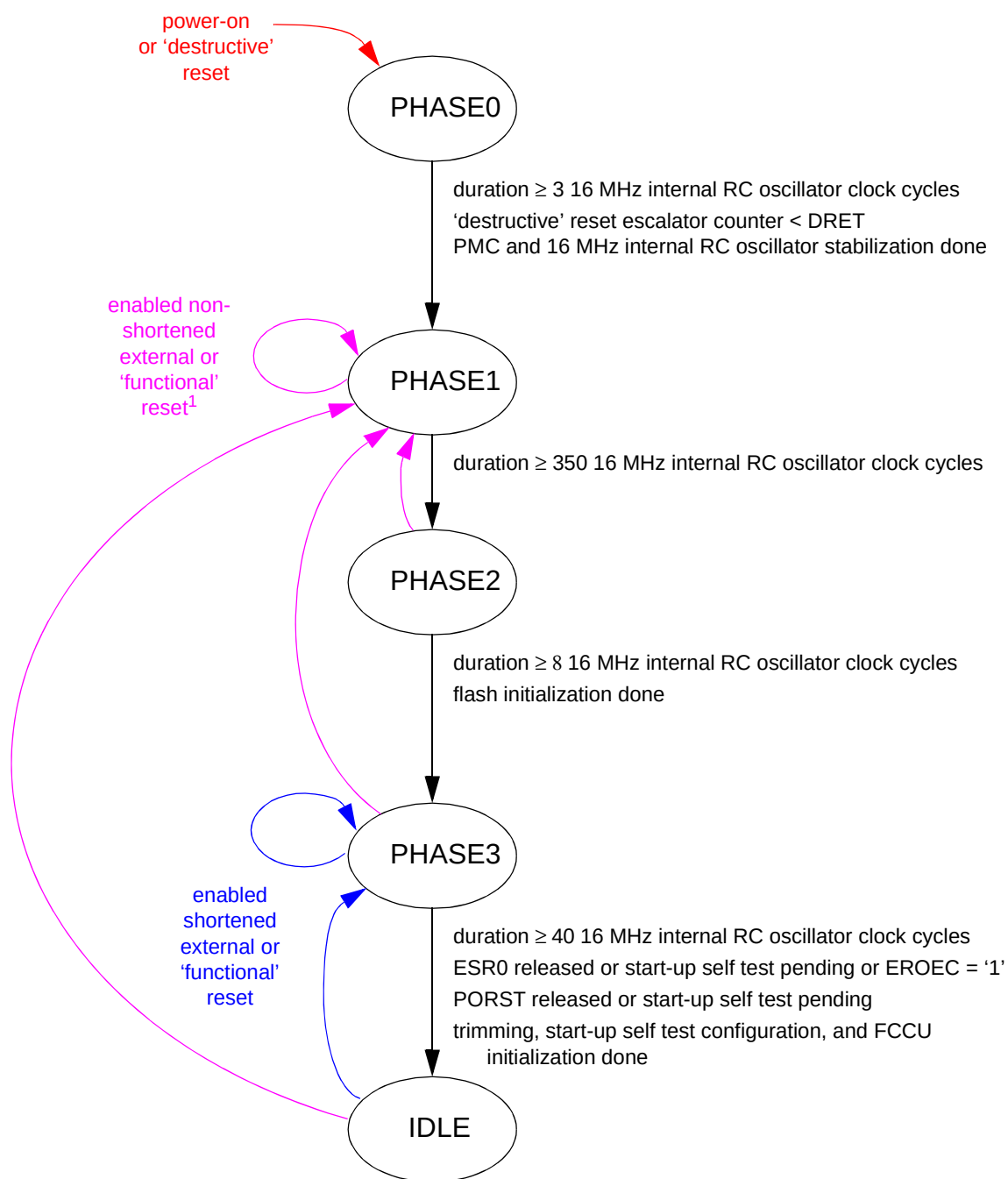


Figure 1413. MC_RGM state machine

57.4.1.1 PHASE0 Phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits PHASE0 and enters PHASE1 on verification of the following:

- all enabled 'destructive' resets have been processed
- all processes that need to be done in PHASE0 are completed
 - PMC and 16 MHz internal RC oscillator stabilization
- a minimum of 3 16 MHz internal RC oscillator clock cycles have elapsed since power-up completion and the last enabled 'destructive' reset event
- the 'destructive' reset escalator counter has not reached the value in the RGM_DRET[DRET] field

57.4.1.2 PHASE1 phase

This phase is entered either on exit from PHASE0 or immediately from PHASE2, PHASE3, or IDLE on a non-masked external or 'functional' reset event if it has not been configured to trigger a 'short' sequence. The reset state machine exits PHASE1 and enters PHASE2 on verification of the following:

- all enabled, non-shortened 'functional' resets have been processed
- a minimum of 350 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened 'functional' reset event

57.4.1.3 PHASE2 phase

This phase is entered on exit from PHASE1. The reset state machine exits PHASE2 and enters PHASE3 on verification of the following:

- all processes that need to be done in PHASE2 are completed
 - flash initialization
- a minimum of 8 16 MHz internal RC oscillator clock cycles have elapsed since entering PHASE2

57.4.1.4 PHASE3 phase

This phase is entered either on exit from PHASE2 or immediately from IDLE on an enabled, shortened 'functional' reset event. The reset state machine exits PHASE3 and enters IDLE on verification of the following:

- all processes that need to be done in PHASE3 are completed
 - trimming, start-up self test configuration, and FCCU initialization
- a minimum of 40 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled, shortened 'functional' reset event
- ESR0 is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence or RGM_EROEC[EROEC] = 1
- PORST is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence

57.4.1.5 IDLE phase

This is the final phase and is entered on exit from PHASE3. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

57.4.2 ‘Destructive’ resets

A ‘destructive’ reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed. Exceptions to the latter are the software ‘destructive’ reset and the external power-on reset which do allow the memory content to be preserved if the chip is configured to not execute a self-test on power-on, ‘destructive’, and external resets.

The status flag associated with a given ‘destructive’ reset event (RGM_DES.F_<destructive reset> bit) is set when the ‘destructive’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

A given ‘destructive’ reset can be optionally disabled by writing bit RGM_DERD.D_<destructive reset>.

Note: The RGM_DERD register can be written only once between two ‘destructive’ reset events.

The chip’s low-voltage detector threshold ensures that, when ‘destructive’ voltage out of range (LVD and/or HVD) is enabled, the supply is sufficient to have the ‘destructive’ event correctly propagated through the digital logic. Therefore, if a given ‘destructive’ reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given ‘destructive’ reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled ‘destructive’ reset will trigger a reset sequence starting from the beginning of PHASE0.

57.4.2.1 External power-on reset—PORST

The bidirectional external power-on reset pin PORST is a special case ‘destructive’ reset.

The detection of a falling edge on PORST will start the reset sequence.

The chip asserts PORST if the reset sequence was triggered by one of the following:

- a power-on reset
- a ‘destructive’ reset event

In this case, PORST is asserted until the end of PHASE0.

The PORST input is disabled immediately as of the PORST output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 µs after the PORST output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

In addition, the reset sequence is not exited into application until PORST is no longer driven low from off-chip. In the case the chip is configured to execute a self-test on power-on, ‘destructive’, and external resets, this is the exit of PHASE3 after self test execution has completed.

57.4.3 External reset

The MC_RGM manages the external reset coming from ESR0. The detection of a falling edge on ESR0 will start the reset sequence.

The status flag associated with the external reset falling edge event (RGM_FES.F_ESR0 bit) is set when the external reset is asserted and the power-on reset is not asserted.

Note: The RGM_FERD register can be written only once between two 'destructive' reset events.

An enabled external reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by the external reset. When RGM_FESS.SS_ESR0 is set, the external reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially when an external reset should not reset the flash memory.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event
- an external reset event if configured via the RGM_FBRE register to assert the external reset
- a 'functional' reset event configured via the RGM_FBRE register to assert the external reset

In this case, ESR0_B is asserted until all conditions for exiting PHASE3 have been met with the exception of the ESR0 assertion check. If the OPT_RGM_ESR0_HW bit of the "UTEST Miscellaneous" DCF client is set to '0', the MC_RGM continues to assert ESR0 until the RGM_EROEC[EROEC] bit has been cleared by software.

In addition, the MC_RGM asserts ESR0 while the start-up self test is being executed.

The ESR0 input is disabled immediately as of the ESR0 output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 μ s after the ESR0 output stops being driven by the chip in order to allow the pull-up on the pin to take effect. If ESR0_B is stopped being asserted by the MC_RGM and the 4 μ s input mask period expires during the reset IDLE phase (e.g., due to the EROEC bit being cleared by software), and now a low value is detected on ESR0, an external reset falling edge event will not occur, and no new reset sequence will be triggered. For a new reset sequence to trigger, ESR0 must first be deasserted.

57.4.4 Functional' resets

A 'functional' reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given 'functional' reset event (RGM_FES.F_<functional reset> bit) is set when the 'functional' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'functional' reset can be optionally disabled by software writing bit RGM_FERD.D_<functional reset>.

Note: The RGM_FERD register can be written only once between two power-on reset events.

An enabled 'functional' reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by a 'functional' reset. When

RGM_FESS.SS_<functional reset> is set, the associated 'functional' reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially in case a 'functional' reset should not reset the flash memory module.

See the MC_ME chapter for details on the STANDBY0 and DRUN modes.

57.4.5 Alternate event generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the RGM_D/FERD and RGM_D/FEAR registers as shown in [Table 1310](#).

Table 1310. MC_RGM alternate event selection

RGM_D/FERD bit value	RGM_D/FEAR bit value	Generated event
0	X	reset
1	0	SAFE mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (i.e., at the reset source that caused the alternate request) and also clearing the appropriate RGM_D/FES status bit.

Note: SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock, and therefore, if the system clock was disabled at the time of the event, the interrupt request is not asserted until the system clock is re-enabled.

Note: If a masked 'destructive' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE0, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

Note: If a masked 'functional' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE1, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

57.4.6 'Functional' reset escalation

'Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' or external resets has occurred between software writes to the RGM_FRET register. This function is enabled by writing a non-zero value to the FRET field of this register.

Once 'functional' reset escalation has been enabled, the MC_RGM increases a counter on each 'functional' or external reset which causes a reset sequence to be initiated (i.e., entrance into PHASE1 or PHASE3 from the IDLE phase). This counter is cleared on a write of any value to the RGM_FRET register and on any power-on or 'destructive' reset. If the

counter reaches the value in the RGM_FRET register's FRET field, the MC_RGM starts a reset sequence at PHASE0 and asserts the F_EDR bit in the RGM_DES register.

57.4.7 'Destructive' reset escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until a new power-on triggers a reset sequence if a number of 'destructive' resets has occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

Once 'destructive' reset escalation has been enabled, the MC_RGM increases a counter on each 'destructive' reset which causes a reset sequence to be initiated (i.e., entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (i.e., entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the value in the RGM_DRET register's DRET field, the MC_RGM enters reset PHASE0 and stays there until the next power-on reset occurs.

57.4.8 Individual peripheral resets

A peripheral may be reset individually without resetting the rest of the chip by setting the appropriate bit in the RGM_PRSTn registers. However, in order to prevent unexpected behavior by the chip, the following sequence must be observed by software.

1. Disable the peripheral that is to be reset via the MC_ME (see the MC_ME chapter for details).
2. Wait for the mode change to complete.
3. Set the bit corresponding to the peripheral in the RGM_PRSTn registers.
4. Wait until the bit corresponding to the peripheral in the RGM_PSTATn registers is set.
5. Clear the bit corresponding to the peripheral in the RGM_PRSTn registers.
6. Wait until the bit corresponding to the peripheral in the RGM_PSTATn registers is cleared.

The peripheral can be kept in reset indefinitely by not executing steps ⁽⁵⁾ and ⁽⁶⁾. After the completion of step ⁽⁶⁾, the peripheral can be enabled again via the MC_ME.

58 Boot Assist Flash (BAF)

58.1 Introduction

The MCU is booted through a collaboration of several blocks, hardware, and firmware. The first boot phases are performed by a state machine inside the System Status and Configuration Module (SSCM). Once completed, the SSCM sends a reset vector to the boot core (the I/O Processor (IOP) on this device) pointing into the Boot Assist Flash (BAF).

The BAF code then checks the life cycle of the device. If it is “Failure Analysis”, BAF enters a loop in which it services the watchdog. BAF does not execute further since the read access to flash memory boot sectors for cores may be disabled if device life cycle is “Failure Analysis”. Otherwise, it searches for a boot header and boots the application code in internal flash memory.

If no boot header is found in internal flash memory, it downloads application code serially using the LINFlexD module. The package pins used by LINFlexD are the same pins used by the M_CAN module. Hence, the external PHY can be either LIN or CAN, allowing the “ASC@CAN” operation. See [Section 58.3.10, UART serial boot configuration](#) for configuration details.

58.2 Memory map

The BAF is located in a 16 KB block of flash that is mapped adjacent to the UTEST flash memory block. The BAF block base address is 0040_4000h. It is one time programmable (OTP) and is programmed during factory test. The address space and memory used by the BAF code are shown in [Table 1311](#).

Table 1311. BAF memory organization

BAF Component	Address
BAF version number	0040_4000h
BAF code entry point	0040_4100h
BAF data area and stack	5280_0000h – 5280_01FFh

58.2.1 BAF image header

BAF image contains a 256-byte header starting from address 0040_4000h. The header contains important information such as BAF version and production disable flag. The BAF header is explained in the following table.

Table 1312. BAF Image Header

Address Offset	Size (bits)	Field Description
0000h	32	BAF Image version
0004h	32	Reserved
0008h	32	Reserved (Part of clock Jitter Activation Constant)

Table 1312. BAF Image Header (continued)

Address Offset	Size (bits)	Field Description
000Ch	32	Clock Jitter Activation Constant
0010h-00FFh	Reserved	

58.2.1.1 BAF image version

The BAF version is a 32-bit field in the image header starting at address 0040_4000h. The 32-bit field is explained in the following table.

Table 1313. BAF image version

Field	Description
[0:7]	Major Number: This field contains the major version number for the BAF image
[8:15]	Minor Number: This field contains the minor version number for the BAF image
[16:23]	BAF Type: This field contains customer specific information.
[24:31]	Reserved

58.2.1.2 Clock Jitter Activation constant

The Clock Jitter Activation constant is a 32-bit flag, but the smallest element in flash that can be programmed is 64 bits, so the first 32 bits are reserved to allow programming of this constant. This 32-bit flag can be programmed by the customer and is used to activate the Clock Jitter feature that adds clock edge jitter to the baud rate clocks of several communications modules. As part of the BAF startup procedure, BAF copies this constant from flash to the PASS Clock Jitter Enable Register.

Table 1314. BAF Clock Jitter Activation Constant

Field	Description
[00:31]	Clock Jitter Activation Constant. The initial state of production disable flag is erased - FFFF_FFFFh. This equates to NO clock jitter.

58.3 Functional description

58.3.1 Boot modes

Depending on the state and the content of its flash memory, the device can enter one of three different boot modes.

- **Boot from internal flash**
If the internal flash contains application code with a valid boot header, the application is booted. The boot header contains flags that enable/disable individual cores and their reset vectors.
- **Serial boot**
If no valid boot header is found and the current lifecycle phase of the device is less than “In Field”, an attempt is made to download the application by a serial protocol using the LINFlexD module. The downloaded code is then executed by the initial boot core.
- **Static mode**
The device enters power safe mode and waits for an external reset.

58.3.2 Device initialization

The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact with the buddy device^(v) and the Hardware Security Module (HSM) in the following way:

1. **Buddy device:** If the device is an emulation device (ED device, buddy device present) and the calibration RAM content is valid, then this RAM is automatically remapped over the calibration flash memory. This is intended to assist with cold-start calibration.
2. **HSM:** If the HSM is enabled and the “wait for HSM” flag is present, then the boot core waits for the HSM to become ready.
3. **BAF sets up the exception vector table** to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors. In the event of an uncorrectable ECC error, BAF issues a destructive reset from the machine check exception handler. BAF programs the Mode Entry module's ME_MCTL[target mode] to issue a destructive reset.
4. **BAF uses 512 bytes of the IOP core's D-MEM area** from 5280_0000h to 5280_01FFh for its stack, data and any code that executes from volatile memory. During the BAF start up procedure, it initializes this section of memory so that there is no ECC error when BAF accesses this memory area.

58.3.3 Flow of control

[Figure 1414](#) shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

v. “Buddy device” refers to a companion device that is used only for system development and debug activities. It is not used in production systems.

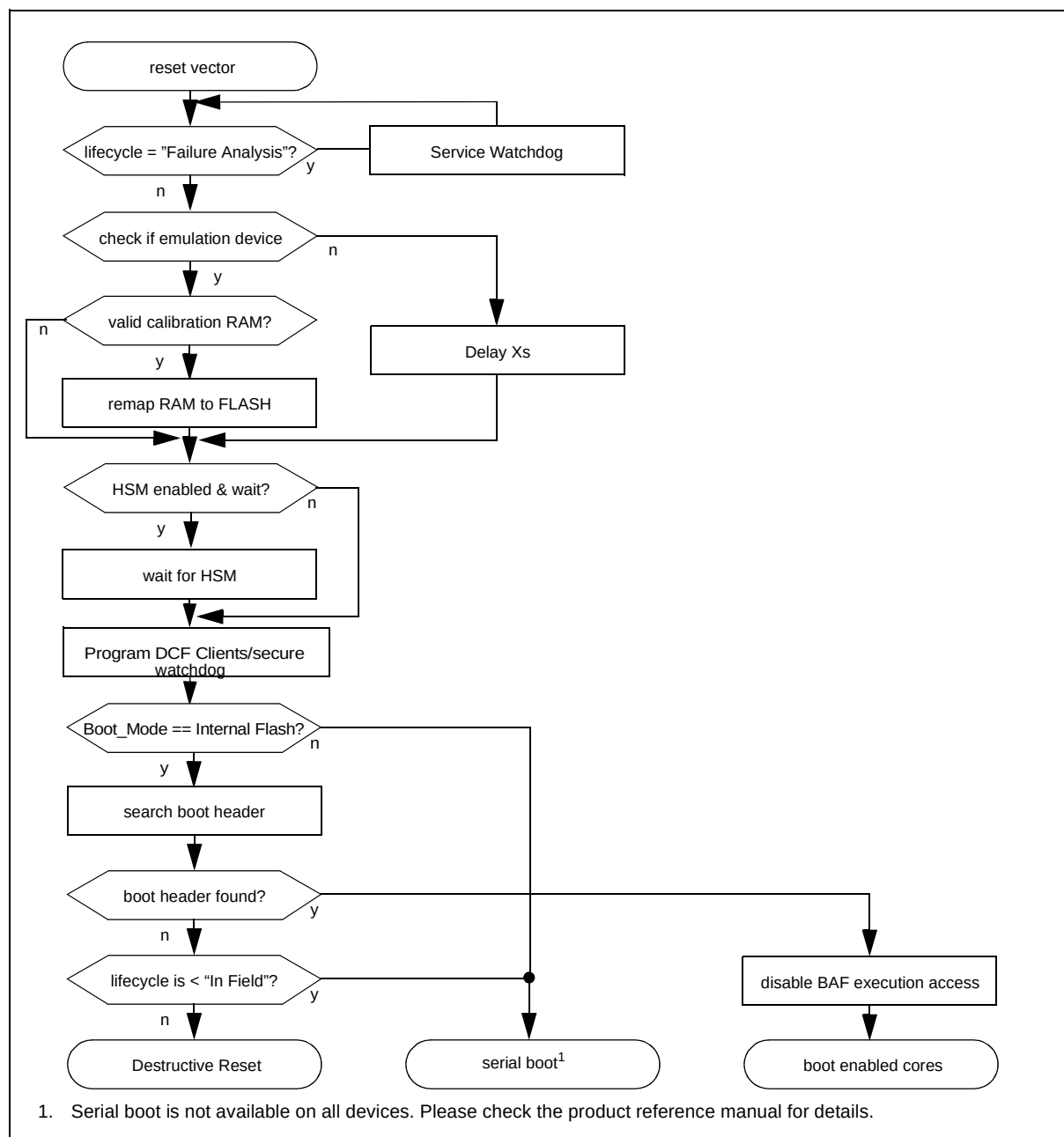


Figure 1414. Flow of control

58.3.4 Optionally remap calibration RAM

If the device is an ED device (buddy device present, `SIUL2_MIDR1[ED] = 1`), and the external overlay RAM has valid content, then this RAM is automatically remapped over selected flash memory regions. The details of the mapping are defined by a 400-byte configuration table at the beginning of the external overlay RAM (0C00_0000h - 0C00_0193h). The layout of the table is depicted in [Table 1315](#).

The content of the overlay RAM is assumed valid:

- if the BD_INIT flag in the initialization status register of the buddy device (BD_SIUL2_ISR) is set,
- and a CRC calculation over the configuration table yields the correct result,
- and the version of the calibration data matches the version of the application software in flash memory ($((\text{uint64_t})\text{ADDR_SWID}) = \text{DATID}$).

In this case, all 32 calibration remap descriptors (PFCRDs) are setup with the values contained in the configuration table. Finally, the PFCRDE field is copied to the enable register in the PFLASH controller.

Table 1315. Remap configuration table

Address Offset (hex)	Name	Contents
00	CRC_REF	CRC reference value
04	ADDR_SWID	Address of 64-bit ID value in SW FLASH
08	DATID	64-bit ID value of the calibration data
0C		
10	PFCRDE	CRD enable flags
14	LSTARTADDR0	PFCRD0, logical start address
18	PSTARTADDR0	PFCRD0, physical start address
1C	CRDSize0	PFCRD0, size
20	LSTARTADDR1	PFCRD1, logical start address
24	PSTARTADDR1	PFCRD1, physical start address
28	CRDSize1	PFCRD1, size
...	...	...
188	LSTARTADDR31	PFCRD31, logical start address
18C	PSTARTADDR31	PFCRD31, physical start address
190	CRDSize31	PFCRD31, size

The CRC calculation is performed using the CRC unit applying the CRC-32 (Ethernet protocol) polynomial. The calculation is initialized with a seed of 0000h. Then, the 100 32-bit words starting at table address offset 04h are moved to the CRC_INP register by the CPU. Finally, the content of the CRC_OUTP register is compared to the CRC_REF value.

58.3.5 Optionally wait for HSM

If the HSM is enabled (SSCM_UOPS[HSE] = 1), the boot core will need to wait for the HSM to become ready (SSCM_UOPS[HSB] = 1). The core then polls the HSM until a ready flag is set. Both SSCM registers can be set by a DCF record. The boot core polls the HSM to HOST FLAGS register (HSM_HSM2HTF) until its least significant bit is set. The boot core also services the watchdog while polling the HSM_HSM2HTF register. This is just in case the HSM is not ready in 16 ms, servicing the watchdog prevents the system going to reset.

58.3.6 Initialization of BAF DCF clients

58.3.6.1 'Soft' DCF clients

DCF clients are 32-bit hardware registers that are written by the SSCM during reset with data that is stored in DCF record format in UTEST region of flash memory. These DCF records are used to configure module registers, and select device operating modes.

The BAF extends the DCF record concept by implementing soft DCF clients. The 64-bit DCF records include:

- 32-bits of data
- 15-bit chip select field (selects the module which implements the DCF clients)
- 15-bit address field (selects a specific DCF client within the selected module)
- Parity bit (not used with UTEST DCF records)
- Stop bit (to indicate the end of DCF record has been reached)

As part of its execution flow, BAF searches through the list of DCF records in UTEST flash to determine if any of the records are to be processed by BAF.

The search starts at one of two addresses: either the start address of DCF records in UTEST (0040_0200h), or the address specified in UTEST at location 0040_00C0h. If either of the two least significant bits of the 32-bit address stored at 0040_00C0h is set, the default start address of 0040_0200h is used. If the two least significant bits are clear, this 32-bit value is assumed to be the start address of the search.

The default setting of the two least significant bits is 11b (erased value of flash memory is FFFF_FFFFh). A user may intentionally program a dummy address with the least two significant bits set in order to disable the search for an alternate address (as the UTEST block is OTP). Valid search start address must start on a 4-byte boundary.

The DCF record search finishes when a DCF record is read with a STOP bit set. The STOP bit may be set because that memory location is not programmed yet (so the list can be added to) or it can be intentionally programmed STOP record to indicate that no more records are added to the list. BAF does not execute the DCF record with STOP bit set. BAF parses the STOP bit before parsing the rest of the records.

BAF uses Chip Select 14 (CS14) for BAF DCF clients. For records with CS14 asserted, the BAF uses the address field (address[14:0]) in DCF records to determine how to process the 32-bits of data. The soft DCF clients programmed by BAF is listed in [Table 1316](#).

The DCF clients which are used for initialization of security watchdog are explained in [Section 58.3.6.2, Security watchdog configuration](#).

A group of four Soft DCF clients is used to provide a mechanism to write configuration information to address locations. The first DCF client receive 32-bit address information. The next three DCF Clients receive the data that is to be written to the address contained in the first client. The three data clients allow 32-, 16- and 8-bit data to be written to the address in the first client. These four clients can be accessed repeatedly to write whole string of data to memory.

Another soft DCF client as stated in [Table 1316](#) provides a simple callback feature. The soft DCF record data contains the start address of a subroutine that the BAF calls before it starts searching for the host boot header and exits to customer application code. The callback routine can reside anywhere in the CPU address space including the BAF code block or UTEST. The callback DCF client can be accessed repeatedly in order to execute a callback sequence.

The serial boot DCF client provides a mechanism using which application can inform BAF to perform serial boot or boot from internal flash. Details can be viewed in section [Section 58.3.6.6, BAF Serial Boot DCF Client -- SER_BOOT_CBACK](#).

Table 1316. BAF DCF client list

Address (hex)	Name	Label	Description
0000h	Security Watchdog Control Register	SEC_SWT_CR	Data to write to SWT Control Register
0001h	Security watchdog Time out register	SEC_SWT_TO	Security watchdog counter value
0002h	Security watchdog service address	SEC_IAC8	Instruction Address compare register value - Code present at this address services the watchdog.
0003h	Security watchdog CPU select	SEC_SWT_CPU	Determines which CPU is to use the security watchdog
0004h...01FFh	Reserved	—	Reserved for future use
0200h	Address	ADDR	The address to which DATA is written
0201h	32-bit Data	DATA32	The 32-bit data that is written to ADDR
0202h	16-bit Data	DATA16	The 16-bit data that is written to ADDR
0203h	8-bit Data	DATA8	The 8-bit data that is written to ADDR
0204h...020Fh	Reserved	—	Reserved for future use
0210h	Callback Address	CALLBACK	The address of a callback function
0211h	Serial boot Callback Address	SER_BOOT_CBACK	The address of serial boot callback function
0212h...7FFFh	Reserved	—	Reserved for future use.

58.3.6.2 Security watchdog configuration

BAF initializes the security watchdog before it jumps to the HOST CPU application. BAF implements four DCF clients for security watchdog initialization. BAF parses the DCF records and looks for configuration data to be used to program the security watchdog registers. BAF first collects all information required to program the security watchdog from DCF records and then program the security watchdog registers. BAF configures only SWT2 as security watchdog for the CPU2 (IOP) core. Configuration of the Security watchdog for other cores has to be performed by the application running on that core as BAF cannot program other core's IAC8 register. Security watchdog detailed configuration steps are explained in the section below.

58.3.6.2.1 Security Watchdog Control Register (SEC_SWT_CR)

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0000h while parsing, BAF programs the DCF client "Security watchdog control register" with the 32-bit value present in the DATA section of the DCF records. Once written, any subsequent writes to this DCF Client are ignored by BAF.

58.3.6.2.2 Security Watchdog Timeout (SEC_SWT_TO)

BAF parses the DCF records. While parsing, if BAF finds a DCF record with CS14 set and address field as 0001h, BAF programs the security watchdog timeout DCF Clients. This is the security watchdog timeout register. BAF checks the value obtained from DCF records. If the value yields a timeout greater than 30 seconds, then BAF changes this value to 30 seconds. Once written, any subsequent writes to this DCF client are ignored.

58.3.6.2.3 Security Watchdog Service Address (SEC_IAC8)

BAF configures the security watchdog in 'fixed address execution service mode'. In this mode, the watchdog is serviced by executing code at the address loaded into the designated IAC8 (Instruction Address Compare) register. If BAF finds a DCF record with CS14 set and address field as 0002h while parsing the DCF records, BAF programs the DCF clients for SEC_IAC8. This DCF client can only be written once. Subsequent writes are ignored.

Warning: To program the IOP core IAC8 special purpose register, the ownership of this register should be with software. If Hardware has its ownership software will not be able to write to it using mtspr instruction.

58.3.6.2.4 Security Watchdog CPU select (SEC_SWT_CPU)

This DCF client indicates to BAF which CPU will use the security watchdog. BAF parses the DCF record and if it finds a record with CS14 set and address field set as 0003h, BAF programs this DCF client. This means that BAF uses the two least significant bits out of the 32-bit data in the DCF record to find the CPU number which will use the security watchdog. [Figure 1415](#) and [Table 1317](#) indicate the data layout for this client in the DCF record. This client is written once during reset. Subsequent writes are ignored.

Note: Due to the limitation that only the IAC8 on the CPU running the BAF code can be accessed, the BAF can only configure the Security Watchdog to operate on CPU2. If another CPU is selected, the Security watchdog is NOT configured

Writing this client causes the BAF to configure the security watchdog based on data written to the three previous DCF clients. If any of the three previous DCF clients are not written, data is configured erroneously.

If security watchdog is assigned to CPU0 or CPU1, relevant CPU IAC8 register is programmed along with other debug configuration registers to enable IAC8 as watch point.

During normal BAF operation, SWT2 is configured to perform a conventional task watchdog function (cause reset if not regularly serviced). If the security watchdog is assigned to CPU2 (IOP), first SWT3 is configured to perform the watchdog function task. The timeout value, configuration, and current time is read from SWT2 and copied to SWT3. Then SWT2 is unlocked with current sequence and enabled as security watchdog.

Figure 1415. SEC_SWT_CPU bit configuration

DCF Client Address 0003h

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				0				0				0			

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				0	0	SC1	SC0

Table 1317. SEC_SWT_CPU field description

Field	Description
[31:2]	Reserved
SC[1:0]	These two bits select the CPU which will use the security watchdog. 00 Reserved 01 Reserved 10 CPU2 IOP Core select 11 Reserved

58.3.6.3 BAF DCF client - ADDRESS

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0200h while parsing, BAF programs this DCF client. This client is written with a 32-bit address from the Data section of DCF record. This 32-bit address is used in conjunction with any one of the DATA32, DATA16 or DATA8 DCF Data clients. BAF does not immediately use the ADDRESS information, but it is stored and used later in conjunction with a DATAX DCF client (DATA32, DATA16 or DATA8). Then BAF writes the data to the address in memory specified by this ADDRESS DCF client.

The user can create a series of data writes to memory by programming a series of DCF records consisting of pairs of records. The first DCF record of the pair writes to ADDR to provide the address. The second DCF record of the pair writes to the DATA client (DATAX) to provide the data. Subsequent pairs of DCF records can be programmed to write to more memory locations. The number of pairs of DCF records that can be programmed is only limited by the amount UTEST space set aside for DCF records.

Figure 1416. BAF DCF ADDR client configuration

DCF Client Address 0200h

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADDRESS[0:15]															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADDRESS[16:31]															

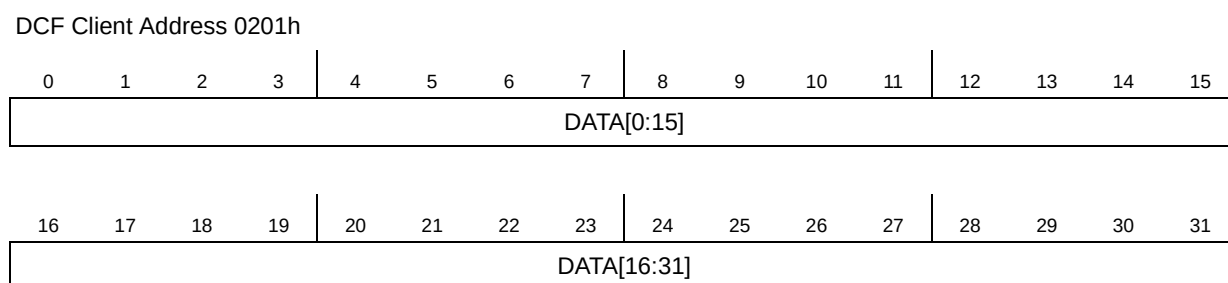
Table 1318. BAF DCF ADDR client description

Field	Description
ADDR[0:31]	32-Bit Address DCF client. This ADDRESS value is used in conjunction with a DATAX value. The BAF writes DATAX value to ADDRESS value in the memory space of the MCU

58.3.6.4 BAF DCF client - DATAX

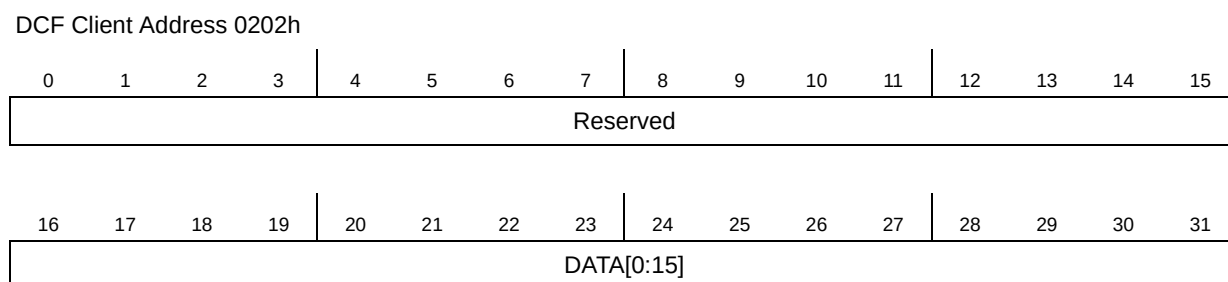
58.3.6.4.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0201h while parsing, BAF programs this 32-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant 2 bits to '0' (Only 32-bit aligned writes are supported) and writes this 32-bit data to the address.

Figure 1417. DATA32 DCF client configuration

58.3.6.4.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0202h while parsing, BAF programs this 16-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant bits to '0' (Only 16-bit aligned writes are supported) and writes least significant 16-bit data to the address. Data in the 16 most significant bits are ignored.

Figure 1418. DATA16 DCF Client Configuration

58.3.6.4.3 DATA8 – 8-bit DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0203h while parsing, BAF programs this 8-bit DCF client. BAF takes the memory address provided by the ADDR client, and writes least significant 8-bit data to the address. Data in the 24 most significant bits are ignored.

Figure 1419. DATA8 DCF Client COnfiguration

DCF Client Address 0203h

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED[16:23]								DATA[0:7]							

58.3.6.5 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0210h while parsing, BAF programs this callback DCF client. This client is written with a 32-bit start address of a function which is invoked by BAF. The BAF code checks the start address is within the UTEST or within the BAF flash memory blocks, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory blocks, no function call is executed. The callback is aborted and BAF moves on to the next DCF records. The function that is called is expected to comply with power.org EABI and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function and no return value is expected.

Warning: Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

BAF executes the callback function as the DCF records are parsed. Therefore several callback functions maybe executed by writing several DCF records that write to the callback DCF clients. The callback functions are executed in the same order as the DCF records.

58.3.6.6 BAF Serial Boot DCF Client -- SER_BOOT_CBACK

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0211h while parsing, BAF programs this serial boot DCF client. This DCF client is written with a 32 bit start address of function which is invoked by BAF. The BAF code checks the start address is within UTEST or within the BAF flash memory block, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory block, no function call is executed. The DCF client execution is aborted and BAF moves on to the next DCF record. The function that is called is expected to comply with power.org EABI and to terminate in orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function.

Warning: Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

The serial boot callback function returns a char value. The returned value informs BAF if serial boot or normal internal flash boot should be executed. For detail regarding return

value of serial boot callback function refer [Table 1319](#).

Table 1319. Serial boot callback function return value definition

Return Values	Description
00h	Boot from internal flash using standard search for boot record. See section 58.3.7 for detail
01h	Serial boot using both CAN and Linflex(in UART mode). For detail see section serial boot
02h-FFh	Reserved

58.3.7 Search boot header and boot options

The BAF searches the flash memory for the boot header, checking the first word location of the 4 × 16 KB blocks and then the 4 × 256 KB blocks. The blocks are searched in the order shown in the following table and once a Boot Header is found, no further blocks are searched.

Table 1320. Locations of boot headers

Search Order	Block	Address
1st	16 KB Code flash memory block1	00FC_0000h
2nd	16 KB Code flash memory block2	00FC_4000h
3rd	16 KB Code flash memory block3	00FC_8000h
4th	16 KB Code flash memory block4	00FC_C000h
5th	256 KB Code flash memory block0	0100_0000h
6th	256 KB Code flash memory block1	0104_0000h
7th	256 KB Code flash memory block2	0108_0000h
8th	256 KB Code flash memory block3	010C_0000h

The first header found that contains the value 005Ah in the first half-word is valid for booting. The whole header structure is shown in [Table 1321](#).

Table 1321. Boot header structure

Address Offset	Contents
00h	Boot Header Configuration (see Figure 1420)
04h	CPU2 Reset Vector
08h	Configuration Bits
0Ch	Configuration Bits
10h	CPU0 Reset Vector
14h	CPU1 Reset Vector
18h	CPUC Reset Vector

Figure 1420. Boot header configuration

Offset 00h

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				0				5				A			

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				CPU1	CPUC	CPU0	CPU2

Table 1322. Boot header configuration field description

Field	Description
CPU1	CPU1 is enabled (starts execution at CPU1 Reset Vector)
CPUC	CPU0-Checker Core is enabled. (starts execution at CPUC Reset Vector. If this is different from CPU0 Reset Vector, then a lockstep error occurs as the CPU comes out of reset)
CPU0	CPU0 is enabled (starts execution at CPU0 Reset Vector)
CPU2 (IOP)	CPU2 (IOP) is enabled (starts execution at CPU2 Reset Vector)

The cores that are enabled in the boot header configuration field are enabled by programming ME_CCTLn[DRUN] register to run in all DRUN modes and their reset vector is set in register ME_CADDRn. Then, the memory area of the BAF is protected from execution for functional safety considerations. This is achieved by disabling BAF execution in PFLASH control register3 (PFLASHC_PFCR3[BAF_DIS]). Finally a mode change (from DRUN to DRUN again) is requested from Mode Entry module to activate the new core configuration and make the cores jump to their respective reset vectors. At this point the BAF has finished its operation.

58.3.8 Production disable activation protocol and implementation

Module failure analysis is a standard requirement. But some OEMs have an additional requirement that upon entering failure analysis mode, the MCU operation is modified in such a way that the MCU can never be used in production ECU. The chosen operational modification for production disable is to apply clock jitters to can modules so that it prevents active communications with another can device on a can network. For detail regarding production disable refer the PASS chapter.

BAF maintains a 64 bit flag in BAF code flash region as explained in [Section 58.2.1.2, Clock Jitter Activation constant](#). On every POR BAF copies the clock jitter activation constant from this flag to clock jitter enable register in PASS module to enable jitters on can baud clock. Default value of this flag is FFFF_FFFFh. Hence jitter on can baud clock is disabled. To enable jitter on can clock the value of this flag should be modified to 7FFF_FFFFh.

BAF flash is write protected and locked. To enable modification of this flag and activate production disable feature, BAF implements production disable activation protocol as explained in [Table 1324 \(Production Disable protocol description\)](#). BAF implements this protocol in a power.ORG EABI compliant callback routine. This callback routine can be invoked by programming a callback based soft DCF client as explained in [Section 58.3.6.5, "CALLBACK – BAF callback DCF client"](#). The callback function address is show in [Table 1323 \(Production Disable Callback routine entry point\)](#)

Table 1323. Production Disable Callback routine entry point

Description	Address
Production disable protocol callback function address	0040_40E0h

Table 1324. Production Disable protocol description

Step	Description	Tool Action/Response	MCU Action/Response
1	After power-on tool holds the MCU in reset and writes the activation code to JDC via JTAG interface. The tool pulls hardware interlock pin high, then releases reset.	Assert hardware interlock pin Write F5A0_AA55h to JDC	Starts production disable activation
2	The MCU responds by writing to the JDC an acknowledgment code and which of the 4 PASS passwords are required to unlock the BAF flash block		Write FA50_000nh to JDC if password for pass group n is required to unlock the BAF flash block. (n is either 0,1,2 or 3)
3	The tool reads the acknowledge from JDC module and writes the relevant password to the JDC module. MCU performs password comparison and writes error code to JDC module password provided by tool was incorrect.	Read JDC Write 256 bit password for pass group n to JDC in eight 32 bit words. (n is 0,1,2 or 3)	Write FA50_F00nh if password does NOT match. Once the status has been read by the tool then cause reset. (n is either 0,1,2 or 3)
4	If more than one password is required to unlock the BAF flash block step 2 and step 3 are repeated until all relevant password has been supplied.		If password matches step 2 and step 3 are repeated for the next pass group until all pass group that lock BAF flash has been processed.
5	The MCU then attempts to program the can jitter activation constant within the BAF flash block. The can jitter activation constant flag is a write once flag. Once written it cannot be re-programmed as BAF flash block is OTP. MCU write a code to JDC to signal success or failure of flash programming operation and issue reset.		Program can jitter activation constant in BAF flash block. If successful write FA51_0000h to JDC. If unsuccessful write FA51_F000h to JDC Issue reset.

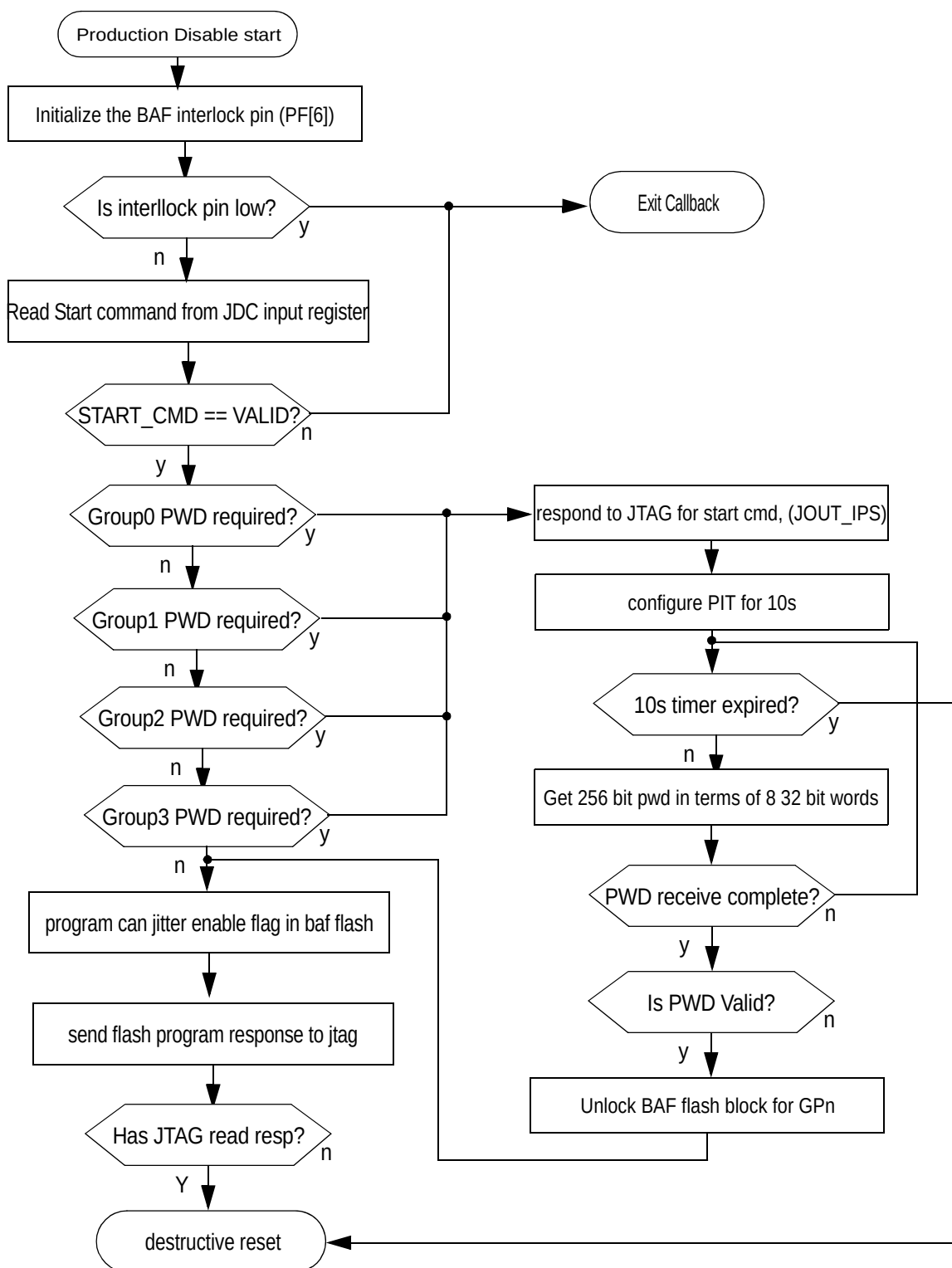


Figure 1421. Production Disable callback function flow chart

58.3.9 Optionally perform a serial boot

If no boot header is found in one of the locations mentioned above or customer selects serial boot using serial boot DCF client, then BAF determines the lifecycle status of the device. If it is not in “In Field” lifecycle, it attempts a serial boot. Otherwise, the boot has failed and the BAF issues a destructive reset.

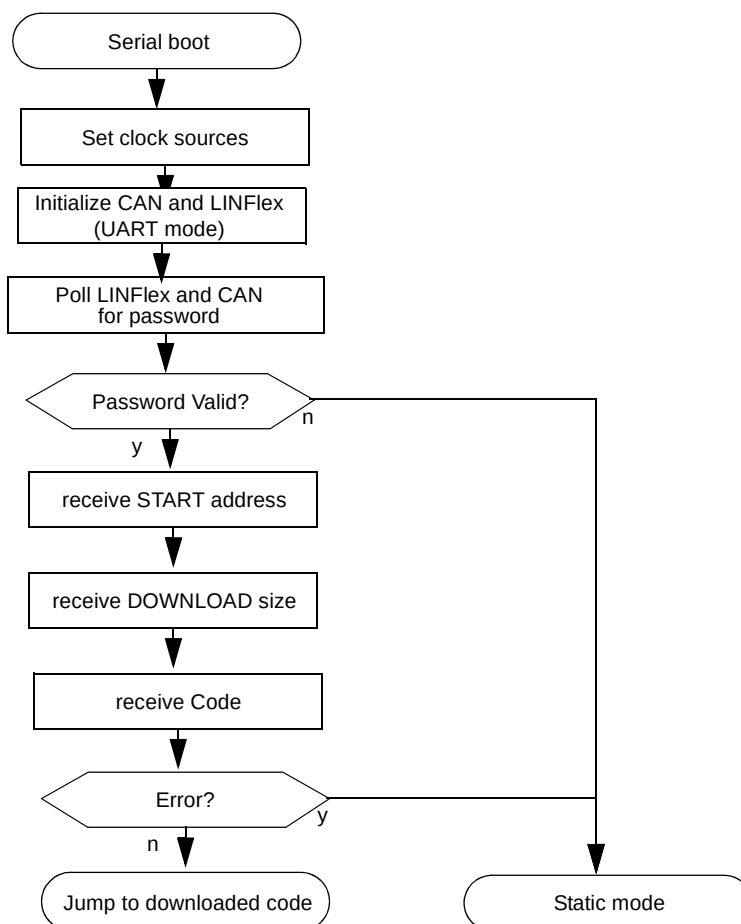


Figure 1422. Flow of control during serial boot

Serial boot allows the download of application to internal sram via linflexD_0 or MCAN_1 interface. After downloading the application image to sram, BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At the start of the process both LinFlexD_0 and MCAN_1 is configured in passive listening mode, connected to the same pair of pads. As soon as either interface receives a valid message, the receiving interface is configured in full Tx and Rx mode. The non receiving interface is switched back to reset state and doesn't participate further in serial boot.

If a CAN message of 8 bytes with ID 11h is received on the MCAN_1 controller first, the BAF program

- Disables the LinFlexD_0.
- Configures LinflexD_0 RXD input mux back to reset state.
- Transitions to CAN serial download protocol routine.

If a CAN message not containing 8 bytes or NOT with ID 11h is received by MCAN controller or a MCAN controller generates a CAN error, it is probable that the tool is sending a Linflex frame. Under such circumstances BAF program

- Ignores the CAN message, if a message arrived.
- Clear any pending CAN error
- Continue monitoring for valid CAN messages.

If any byte from Linflex is received first the BAF program

- Checks if the first byte is FEh of the password. If not continue polling. The Data may be a valid CAN message.
- If it is first byte of the password (FEh) then disable MCAN_1 controller and restore to its reset state.
- Configures MCAN_1 pads to their reset state.
- Transition to Linflex (Uart mode) serial boot

58.3.9.1 Serial boot mode protocol

From a high-level perspective, the download protocol follows steps:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes and VLE bit^(w) to MCU.
3. Host transmits a sequence of data bytes that are to be executed.
4. CPU2 (IOP) executes code from start address provided in step 2.

Each step must be complete before the next step starts.

The exact protocol between LinFlex and CAN are slightly different.

Communication is performed in half-duplex mode, any transmission from the host is followed by the MCU transmission.

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received data to the host.
3. The host verifies whether the echo is correct.
 - If the data is correct, the host can send the next byte of data.
 - If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

w. Since this device supports only VLE code and no Book E code, this flag is used only for backward compatibility

58.3.9.2 Linflex download protocol

The communication is performed in half-duplex mode, Any transmission from the host is followed by the MCU transmission.

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received data to the host.
3. The host verifies whether the echo is correct.
 - If the data is correct, the host can send the next byte of data.
 - If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

The tool sends data in the following order:

1. 8 bytes that represent the password.
2. 4 bytes that represent the start address where the code is to be downloaded.
3. 4 bytes that represent the number of bytes to be downloaded and the VLE bit.
4. x bytes of code that is downloaded to SRAM and executed (x must match the value specified in step ⁽³⁾).

58.3.9.3 MCAN download protocol

The MCAN download protocol uses very similar structure to the LINFlex protocol, but it makes use of the message ID's provided by CAN.

The tool sends data packed into standard CAN message in the following manner:

1. A message with 11h as the ID and an 8-byte length to send the password. The MCU transmits the same message, but with an ID of 1h.
2. A message with 12h as the ID and an 8-byte length to send the start address, size of the code and VLE mode bit. The MCU transmits back the same data but with an ID of 2h.
3. A message with 13h as the ID is used to send data that is to be downloaded to SRAM. The MCU transmits the same data with an ID of 3h.

58.3.9.4 Download 64-bit password and password check

The first 64 bits received represent the password. The MCU only supports public passwords. The received password is always compared with the public password FEED_FACE_CAFE_BEEFh.

- If the correct password is received, BAF continues its task.
- If an incorrect password is supplied, BAF puts the device into static mode (low power safe mode).
- If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU.

58.3.9.5 Download start address, VLE bit and code size

After the password, the next 8 bytes received by the MCU contains a 32-bit start address, the VLE mode bits, and 31-bit code length as shown in [Figure 1423](#) and [Figure 1424](#)

The VLE bit (Variable Length Encoding) is used to indicate for which instruction set the downloaded code is compiled. The UART serial boot protocol supports download of VLE Code (VLE = 1) and Book E Code (VLE = 0).

The MCU only supports VLE code. If the BAF detects that the host is sending Book E code, the BAF puts the device into static mode.

The start address defines where the received data is stored and where the MCU branches after the download is finished. The three LSB bits of the start address are ignored by BAF, such that the load address is 64-bit double-word aligned.

The length determines how many data bytes has to be loaded.

Figure 1423. Start address

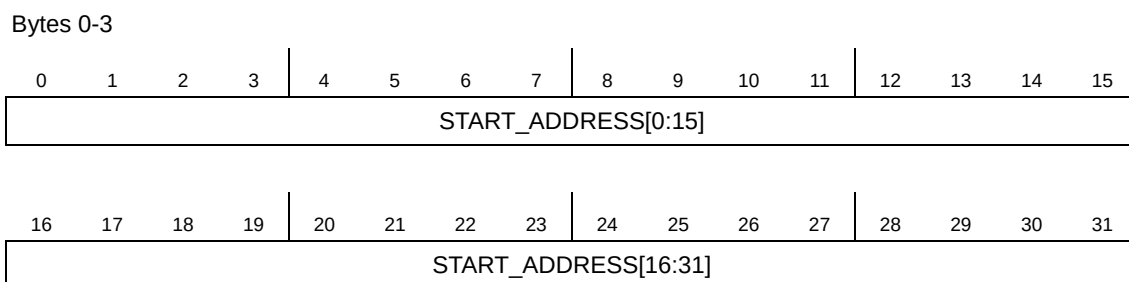


Figure 1424. Download VLE bit and code size



58.3.9.6 Download data

Each byte of data received is downloaded into device SRAM starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until a number of bytes of data received matches the code length specified in the previous protocol step.

Since the SRAM is protected by 64-bit wide Error Correction Code (ECC), BAF always writes to SRAM grouped in 64-bit double-words. If the last byte does not fall into a 64-Bit bounder, the BAF fills it with zero's and renders the data aligned on a 64-bit bounder. After the last data is written to SRAM, the BAF writes a dummy double-word (0000_0000_0000_0000h) to the next address location. This is written to avoid possible ECC error during core prefetch.

58.3.9.7 Execute code

The BAF program waits for the last echo message transmission to be completed. Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which

was received in step 2 of the protocol. At this point the BAF has finished its tasks and the MCU is controlled by new code executing from SRAM.

58.3.10 UART serial boot configuration

58.3.10.1 Configuration

Boot using UART protocol is implemented by LINFlexD_0 module. The pins used are:

- LINFlexD_Tx, corresponding to pin PA[10]
- LINFlexD_Rx, corresponding to pin PA[11]

When Serial Boot mode is started, pin PA[11] (LINFlexD_Rx) is configured as an input and multiplexed to the LINFlexD_0 module. The pin PA[10] (LINFlexD_Tx) remains in a passive input state until the first byte of protocol is received. It is then configured as an output and multiplexed to LINFlexD_0 module. This is done because it is only certain that the application expects pin PA[10] to be an output when a serial message is received on pin PA[11]. The application may not be using the Serial Boot Mode function and may be using pin PA[10] as an input pin with some external hardware driving the pin. Inadvertently driving this pin as an output would cause contention and possible damage.

The LINFlexD module operates in half-duplex mode. The interface is either transmitting a byte or receiving a byte, but never both at the same time. It is expected that LINFlexD pins are connected to a physical layer bus driver/receiver (PHY). Many PHYs enforce half-duplex mode by using a single data channel for both transmitting and receiving data (e.g. LIN, CAN, FlexRay). The side effect of this is that all data transmitted on Tx pin is also received on Rx pin. To prevent the LINFlexD module receiving its own transmission, the Rx pin is blocked during Tx Pin transmission.

The LINFlexD controller is configured to operate at a baud rate of $f_{\text{xosc}} / 833$, using 8-bit data frame without parity bit and 1 stop bit.

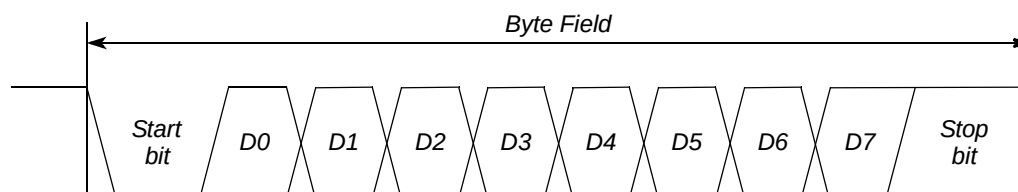


Figure 1425. LINFlexD bit timing in UART mode

58.3.10.2 MCAN Configuration

The MCAN_1 controller is configured to operate at a baud rate equal to crystal frequency divided by 40, using the standard 11 bit identifier format detailed in CAN 2.0A specification. See [Table 1326](#) for Baud rates.

The Bit timing is configured as shown in [Figure 1426 \(CAN Bit Timing\)](#)

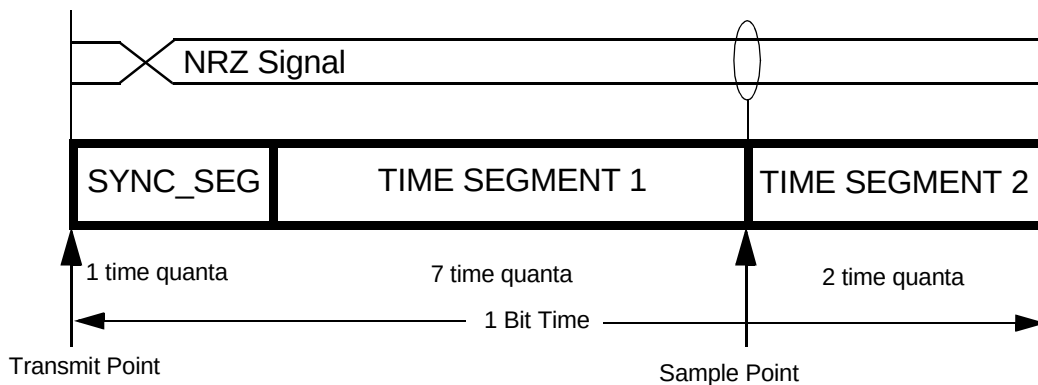


Figure 1426. CAN Bit Timing

58.3.10.3 LinflexD_0 and MCAN_1 pin configuration

[Table 1325](#) shows the device pin configurations for LINFlexD_0 and MCAN_1 modules.

Table 1325. MCAN_1 and LinFlexD_0 pin configuration

Pads	Reset Function	Initial Serial Boot Mode		Serial Boot Mode after a valid CAN message received	Serial Boot Mode after a valid LINFlex message received
		Function	Pad Configuration	Function	Function
PA[10]	GPIO Input buffer off Output buffer off	MCAN_1 Tx	Output:— Push/Pull medium drive no pullup/pulldown	MCAN_1 Tx	LINFlexD_0 Tx
PA[11]	GPIO Input buffer off Output buffer off	LINFlexD_0 Rx and MCAN_1 Rx	Input:— pull-up hysteresis TTL voltage level	MCAN_1 Rx	LINFlexD_0 Rx

58.3.10.4 Baud rates

The LINFlexD baud rate is set by frequency of external crystal oscillator/833.333.

[Table 1326](#) shows some possible baud rates.

The MCAN baud rate is set by the frequency of the external crystal/40.

Table 1326. LinFlex_0 and MCAN_1 baud rates

Crystal frequency	LINFlex baud rate	MCAN baud rate
8 MHz	9,600 baud	200 kbaud
12 MHz	14,400 baud	300 kbaud
16 MHz	19,200 baud	400 kbaud
20 MHz	24,000 baud	500 kbaud
40 MHz	48,000 baud	1000 kbaud

58.3.10.5 Protocol summary

[Table 1327](#) summarizes the protocol and BAF action during this boot mode.

Table 1327. UART serial boot mode download protocol

Protocol steps	Host message sent	BAM response message	Action
1	64-bit password (MSB first)	64-bit password (MSB first)	Password checked for validity and compared with stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + 31 bit number of bytes (MSB first)	VLE bit + 31 bit number of bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	8 bits of raw binary data	8 x 8 bits of data are packed into 64-bit double-words. These double-words are stored in SRAM starting from the Load address. The Load address increments until the number of data received and stored matches the size as specified in the previous steps.
5	none	none	Branch to downloaded code

58.4 Resources

- **Clocks**
Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before execution of the application code (in flash memory or received via serial download).
- **LINFlexD**
The LINFlexD module is only used if normal boot header search fails and a serial download boot is attempted. All settings are restored before the application code in system RAM is executed.
- **CRC**
The CRC module is only used to validate external overlay contents if the device is an ED device and the buddy device has been initialized.
- **PFLASH**
Remapping of the external overlay memory is configured automatically under certain conditions (see [Section 58.3.4, Optionally remap calibration RAM](#))
- **Interrupts**
No external interrupts are enabled or generated by the BAF. BAF, however, handles machine check exceptions with MSR[ME] bit disabled.
- **Security watchdog (SWT3)**
When SWT2 is chosen as security watchdog, SWT3 is used for Task monitoring purpose.

59 System Status and Configuration Module (SSCM)

59.1 Introduction

59.1.1 Overview

The System Status and Configuration Module (SSCM) is pictured in [Figure 1427](#).

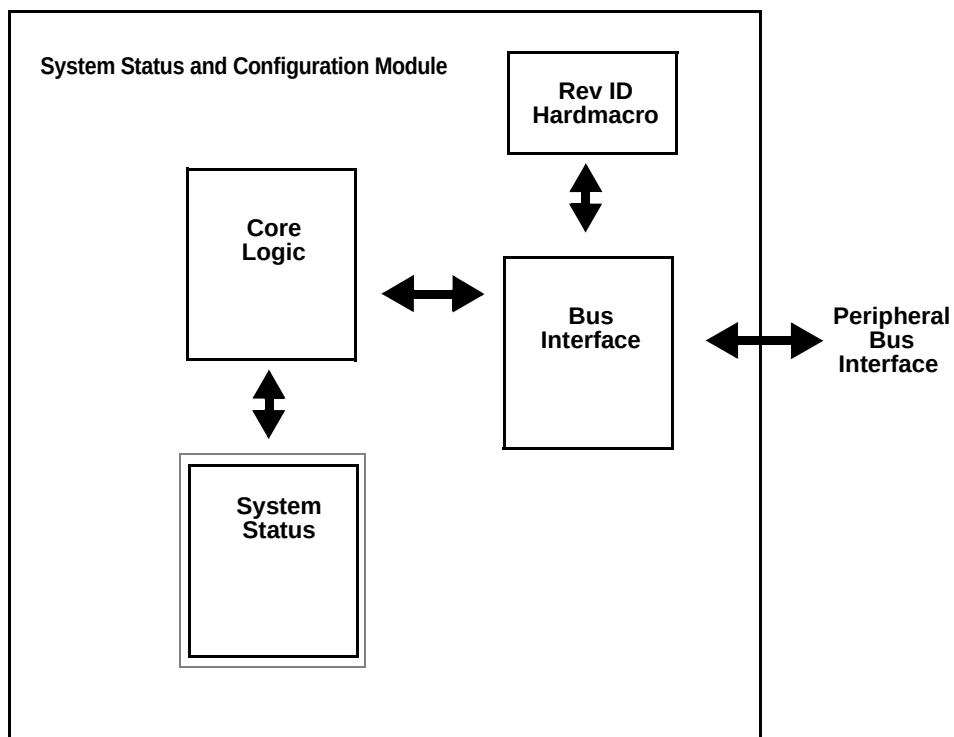


Figure 1427. SSCM block diagram

59.1.2 Glossary

Table 1328. Glossary

Term	Definition
0011b	Binary numbers
0Fh	Hexadecimal numbers
Pin	External physical connection.
Signal	Electronic construct whose state or change in state conveys information.
x	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded X001b, the state of the first bit is a don't care.
y	y is used for a place holder of 1 hex digit for unknown values.

59.1.3 Features

The SSCM includes these distinct features:

- System Configuration and Status
 - Device Mode and System Status
 - Determine primary boot vector
 - Determine HSM boot vector

59.1.4 Modes of operation

The SSCM operates identically in all system modes.

59.2 External signal description

The SSCM has no external pins.

59.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

[Table 1329](#) shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

Table 1329. SSCM memory map

Offset from SSCM_BASE	Register	Access	Reset Value	Section/ page
0000h	SSCM System Status (STATUS)	R/W	yyy0h ⁽¹⁾	on page 2574
0002h	SSCM System Memory and ID Register (MEMCONFIG)	R	yyyyh ⁽¹⁾	on page 2575
0004h	Reserved	—	—	—
0006h	SSCM Error Configuration Register (ERROR)	R/W	0000h	on page 2575
0008h	Reserved ⁽²⁾	—	—	—
000Ah–001Ch	Reserved	—	—	—
0020h	SSCM HSM and User Option Status Register (UOPS)	R	yyyy_yyyh ⁽¹⁾	on page 2576
0024h	Reserved	—	—	—
0028h	Processor Start Address Register (PSA)	R	yyyy_yyyh ⁽¹⁾	on page 2577
002Ch	Reserved	—	—	—
0030h	SSCM HSM Start Address Register (HSA)	R	yyyy_yyyh ⁽¹⁾	on page 2578
0034h	Life Cycle Status Register (LCSTAT)	R		
0038h–0048h	Reserved	—	—	—

1. See register description for reset value details.

2. No register abort on this location.

59.3.1 Register descriptions

The following registers are available in the SSCM. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits should be masked out during any read/write operations to avoid conflict with future revisions.

59.3.1.1 System Status Register (STATUS)

The System Status register reflects the current state of the system.

Figure 1428. System Status Register (STATUS)

Offset 0000h											Access: Read/Write					
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CER	0	NXEN1	NXEN	1	0	0	BMODE			VLE	0	0	0	0
W		w1c														
Reset	0	0	0	_(2)	_(2)	1	0	0	_(2)			_(1)	0	0	0	0

1. Reset value depends on the associated option bits in flash memory.
2. Reset value depends on the device status after leaving reset.

Table 1330. STATUS field descriptions

Field	Description
CER 1	Configuration Error. This field indicates that the SSCM has detected a configuration error. This error may have been detected during the reset sequence. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct If a non-permanent (transient) error is detected, this bit can be cleared by writing a 1. If the error is permanent, the bit will reappear immediately.
NXEN1 3	Processor 1 Nexus enable status. 0 Processor 1 Nexus disabled. 1 Processor 1 Nexus enabled
NXEN 4	Processor 0 Nexus enable status. 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
BMODE 8:10	Device Boot Mode. 101 Boot from BootFlash All other values are reserved. This field is only updated during reset.

Table 1330. STATUS field descriptions (continued)

Field	Description
VLE 11	Variable Length Instruction Mode. When booting in Single Chip (SC) mode, this field indicates that the code stored in the flash memory is using the VLE instruction set. The value of this field is determined by the RCHW field of the flash memory boot sector. 0SC Boot Location contains standard PPC code 1SC Boot Location contains VLE code
MBE0 14:15	Number of multi-bit (uncorrectable) ECC errors detected during device configuration. If more than 3 errors are detected, the device will automatically reset itself and attempt configuration again.

59.3.1.2 System Memory and ID Register (MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG ID.

Figure 1429. System memory and ID register (MEMCONFIG)

Offset 0002h												Access: Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	JPIN										1	MREV				0
W																
Reset	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	1	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0

1.

Reset value reflects the JTAG ID of the device.

2.

Reset value depends on boot mode and security status (see SSCM Device Configuration chapter).

Table 1331. MEMCONFIG field descriptions

Field	Description
JPIN 0:9	JTAG Part ID Number
MREV 11:14	Minor Mask Revision

59.3.1.3 Error Configuration Register (ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Figure 1430. Error Configuration Register (ERROR)

Offset 0006h												Access: Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PAE	RAE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1332. ERROR field descriptions

Field	Description
PAE 14	Peripheral Bus Abort Enable. This bit enables bus aborts on any access to an off-platform peripheral slot that is not used on the device. This feature is intended to aid in debugging when developing application code. 0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception
RAE 15	Register Bus Abort Enable. This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. This feature is intended to aid in debugging when developing application code. 0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception <i>Note: Transfers to peripheral bus resources may be aborted even before they reach the peripheral bus (in example, at the AIPS level). In this case, the PAE and RAE register bits have no effect on the abort.</i>

59.3.1.4 HSM and User Option Status Register (UOPS)

Figure 1431. HSM and User Option Status Register (UOPS)

Offset 0020h												Access: Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	HSB		HSE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾	0/1 ⁽¹⁾

1. Values are determined by DCF record and life cycle - default is 0.

Table 1333. UOPS field descriptions

Field	Description
HSB 28:30	HSM Boot Configuration 000 - When the device is in BAF mode, BAF will not wait for HSM ready flag before allowing application code to start. In SC mode the SSCM will not wait for HSM ready flag before allowing application code to start. 001 - When the device is in BAF mode, BAF waits for HSM ready flag before allowing application code to start. In SC mode the SSCM waits for HSM ready flag before allowing application code to start. All other combinations are reserved. These bits can be set via DCF record (see HSM documentation).
HSE 31	HSM Enabled - this bit is set to 1 if HSM is enabled via DCF record

59.3.1.5 Processor Start Address Register (PSA)

Figure 1432. Processor Start Address Register (PSA)

Offset 0028h												Access: Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SADR															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADR															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Table 1334. PSA field descriptions

Field	Description
SADR 0:31	Start Address - the boot processor will start executing application code from this address. In SC mode this indicates the location determined by the RCHW search. In BAF mode this will show the start address of the BAF code.

59.3.1.6 HSM Start Address Register (HSA)

Figure 1433. HSM Start Address Register (HSA)

Offset 0030h												Access: Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HADR															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HADR															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Table 1335. HSA field descriptions

Field	Description
HADR 0:31	HSM Start Address - HSM will start executing application code from this address. The value of this register is determined by the HSM boot header search.

59.3.1.7 Life Cycle Status Register (LCSTAT)

Figure 1434. Life Cycle Status Register (LCSTAT)

Offset 0034h												Access: Read Only				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0/1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	PLC		0	0	0	0	0	LC		
W																
Reset	0	0	0	0	0	0	0/1 ⁽¹⁾ 0/1 ⁽¹⁾		0	0	0	0	0	0/1 ⁽²⁾ 0/1 ⁽²⁾ 0/1 ⁽²⁾		

1. Reset value reflects the Pre-Life Cycle of the device.

2. Reset value is the Life Cycle of the device.

Table 1336. LCSTAT field descriptions

Field	Description
PLC 22:23	Pre-Life Cycle - as determined by the SSCM by reading the PLC slots. The encoding is: JDP_PROD = 01 IN_FIELD = 11 FAIL_ANALYSIS = 00
LC 29:31	Life Cycle - as determined by the SSCM by reading the LC slots. The encoding is: CUST_DELIV = 011 PROD_OEM = 010 IN_FIELD = 111 FAIL_ANALYSIS = 000

59.4 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

59.4.1 DCF mechanism

The SSCM maintains three DCF record areas - two in TestFlash and one in UTEST or Shadow Flash. The first TestFlash area is for differentially stored manufacturer data, the second for normally stored manufacturer data. The UTEST or Shadow area is for customer-controller data.

Which of those areas are scanned and where they are located is controlled via SSCM parameters - consult the SSCM integration guide for details.

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device will process these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

Each record is 64 bits in length. [Table 1337](#) shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records should be processed.

Table 1337. DCF start record

0:31
05AA 55AFh
0:31
0000 0000h

The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - See [Table 1338](#). In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Table 1338. Format of a DCF data record

0:31			
WDATA[31:0]			
0:14	15:29	30	31
CS[14:0]	ADDR[16:2]	PRTY	STOP

The stop record indicates that processing should stop. The general format of the stop record is shown in [Table 1339](#). Only the STOP bit needs to be 1 in order to form a stop record – All other bits are ignored. An unprogrammed location in UTEST flash will be interpreted as a stop record.

Table 1339. DCF stop record

0:31	
rsvd	
0:30	31
rsvd	1

If n data records are to be stored in UTEST, the data structure must be as shown in [Table 1340](#).

Table 1340. Series of DCF records in UTEST

ADDR offset	DATA			
00h	05AA 55AFh			
04h	0000 0000h			STOP=0
08h	WDATA[31:0]			
0Ch	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
10h	WDATA[31:0]			
14h	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
$8n - 1 + 0h$	reserved			
$8n - 1 + 4h$	reserved			1
$8n + 0h$				
$8n + 4h$				

There must never be an unprogrammed record in the data structure, as that would be interpreted as a stop record, so subsequent records would be ignored. This allows programming the records in several sessions, each time appending new records at the end of the list, as shown in [Figure 1435](#)

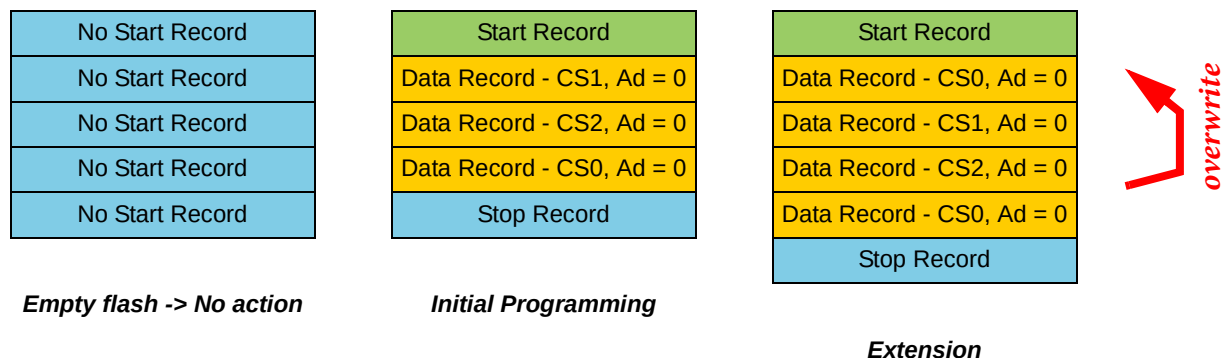


Figure 1435. Appending DCF records

In this case a record may overwrite a client's value which was already set by a previous record. However whether such an operation is allowed depends on the rules given for the client - some clients are write-once, only allow setting bits but not clearing etc.

The parity bit (PRTY) is required for some DCF records. The parity scheme used is even parity, so the number of 1s in the WDATA and PRTY fields needs to be even. (For example, if the WDATA field has the value 0000_0001h then the PRTY field needs to be set to 1, so that the total number of 1s is even.)

59.4.2 ECC error monitoring

During DCF scanning, the SSCM will monitor the flash status signals for ECC and other unexpected access errors. If such an error occurs the SSCM will request a reset.

59.4.3 Boot mode functionality

The device supports the following boot modes for the main boot core:

- Boot Assisted by Flash (BAF) - the device will boot from TestFlash, (interface with the HSM if available) and perform device setup

If booting is not possible with the selected configuration (For example, no Boot ID is found in the selected boot location) then the device will enter serial boot mode.

If the HSM is enabled via DCF record, the HSM will boot from the location found via the HSM boot header search. [Section 59.4.5, HSM boot header search](#).

59.4.4 BAF configuration

By default the device will boot from BAF.

In BAF mode the boot core will start to execute the BAF code at the location shown in the SSCM section of the "Device Configuration" chapter. The BMODE field in the SSCM status register will be set to the value 101b. Details of the BAF operation are described in the BAF chapter.

59.4.5 HSM boot header search

The SSCM searches the code flash memory of the HSM module for a valid boot header (see the SSCM section of the “Device Configuration” chapter for address(es)) and provides the boot vector to the CPU of the HSM module.

The HSM Boot Header has the format as described in [Table 1341](#).

Table 1341. Boot header structure

00h	Boot Header ID	
08h	Reserved	Start Address
10h	Reserved for Configuration Bits	

A valid Boot Header ID has the value FFFF_0000_FFFF_0000h.

The first valid Boot Header found is the one which is passed to the HSM CPU. In order to switch from one Boot Header to a Boot Header at a higher address, the Boot Header at the lower address must be invalidated. This can be achieved by over-programming the Boot Header ID location with 0000_0000_0000_0000h. Since this value results in the same ECC encoding it will still be possible to read the location without causing an ECC error.

59.4.6 Life Cycle

The SSCM will determine the Life Cycle of the device by reading the Life Cycle slots from the UTEST flash area. The read operation is done during the reset phase with normal timings and it is protected by both operating monitors and ECC check. In addition a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- JDP Production
- Customer Delivery
- OEM Production
- In Field
- FA

The LC is written into 5 slots, 128 bits each, at fixed locations in the UTest Flash block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- the valid field
- the invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 statuses:

Table 1342. Life Cycle Slots

LC Slot		LC slot value
Valid Field (64 Bits)	Invalid Field (64 Bits)	
Erased	Erased	Erased
Marked	Erased	Active

Table 1342. Life Cycle Slots

LC Slot		LC slot value
Valid Field (64 Bits)	Invalid Field (64 Bits)	
Marked	Marked	Inactive
Any other values		Illegal

“Marked” in this case means that the value has been programmed with the bit pattern 55AA_50AF_55AA_50AFh, “Erased” is detected by the bit pattern FFFF_FFFF_FFFF_FFFFh. [Table 1343](#) shows how the slots are arranged in memory.

Table 1343. Life Cycle Slots in Memory

Offset	LC Slot word
00h	Valid[31:0]
04h	Valid[63:32]
08h	Invalid[31:0]
0Ch	Invalid[63:32]

The Life Cycle is determined as shown in [Table 1344](#) (A “*” represents a “don’t care” condition, “Erased” means the slot returns all 1’s). The priority of the entries is from top to bottom, so the first row which applies will determine the resulting Life Cycle.

Table 1344. Life Cycle Slots

PLC = JDP_PROD	LC Slot 0 JDP_PROD	LC Slot 1 CUST_DEL	LC Slot 2 OEM_PROD	LC Slot 3 IN_FIELD	LC Slot 4 FA	Resulting Life Cycle
*	Active	Erased	Erased	Erased	Erased	JDP Production
*	<i>Inactive</i>	Active	Erased	Erased	Erased	Customer Delivery
*	<i>Inactive</i>	<i>Inactive</i>	Active	Erased	Erased	OEM Production
*	<i>Inactive</i>	<i>Inactive</i>	<i>Inactive</i>	Active	Erased	In Field
*	<i>Inactive</i>	<i>Inactive</i>	<i>Inactive</i>	<i>Inactive</i>	Active	FA
No	Erased	Erased	Erased	Erased	Erased	System Reset
No	*	*	*	*	Erased	In Field
No	*	*	*	*	Not Erased ⁽¹⁾	System Reset
Yes	*	*	*	*	*	JDP Production

1. Any value but FFFF_FFFF_FFFF_FFFFh

59.5 Initialization and application information

59.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (see [Section 59.3.1, Register descriptions](#)).

59.6 Additional safety measures

59.6.1 Spurious reset protection

The SSCM implements protection against spurious module resets (i.e. resets which only effect the SSCM but not other modules) by gating the state machines with the expected status of the RGM. In case of a spurious reset the SSCM will not interfere with the Flash bus or overwrite configuration registers. Status internal to the SSCM will be lost, however.

60 Power Management Controller digital interface (PMC_dig)

60.1 Introduction

The power management controller (PMC) consists of two blocks: an analog block and a digital block. This chapter describes the digital block of the PMC.

The PMC_dig block contains all of the registers and digital logic to generate all of the enable signals, trim control bits, POR reset generation, and test logic for the analog block. The PMC digital logic also contains the controls for PMC analog outputs to be sensed with the ADC module.

In addition, the PMC digital logic contains the enable signals, trim bits, and other registers for the temperature sensor.

There are also several custom interfaces to various other blocks: The Reset Generation Module (MC_RGM) block receives the LVD values during POR to use for phase transition conditions.

There is also a custom interface to flash memory via the SSCM (and DCF client blocks) that is used to load the trim values during initial POR.

The TESTPIN must never be asserted to assure that the LVD/HVDs are active in the system.

The PMC digital block generates a transfer error event to the system if the selected address is greater than the highest address in the PMC digital memory map. A transfer error is not generated if an unused address within the PMC digital memory space is accessed.

Note: It is not valid to leave any of the power domains measured by an LVD or HVD unpowered. They must always be connected in such a way as to be in a valid operating range to get the device out of POR.

60.2 Block diagram

The block diagram for the PMC is shown in [Figure 1436](#).

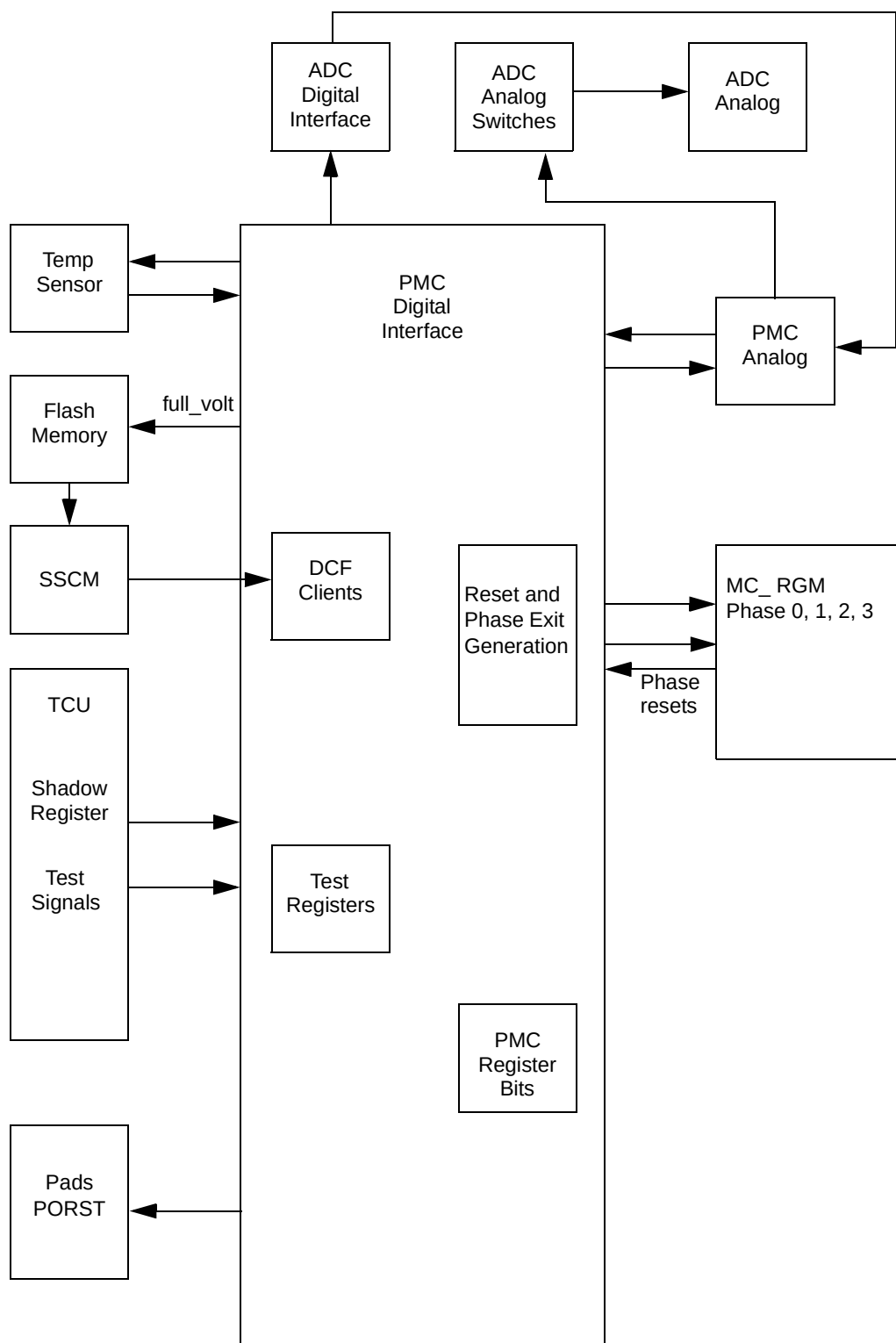


Figure 1436. Digital PMC block diagram

60.3 Memory Map/Register Definition

Table 1345 shows the PMC memory map. The PMC includes many registers for configuring and monitoring LVD monitors, some other control registers.

Table 1345. Power Management Controller Memory Map

Address	Register	Access	Reset Value	Section /Page
PMC_BASE + 0x0000	Supply Gauge Status Register (GR_S)	R	0x0000_0000	60.3.1.1/2589
PMC_BASE + 0x0004	Pending Gauge Status Register (GR_P)	R	0x0000_0000	60.3.1.2/2591
PMC_BASE + 0x0008	Interrupt Enable Pending Register (IE_P)	R/W	0x0000_0000	60.3.1.3/2594
PMC_BASE + 0x0030	LVD108 Event Pending register VD3 (EPR_VD3)	R/W	0x0000_0000	60.3.1.4/2597
PMC_BASE + 0x0034	Reset Event Enable VD3 (REE_VD3)	R/W	0x0000_00C1	60.3.1.5/2599
PMC_BASE + 0x0038	Reset Event Selection VD3 (RES_VD3)	R/W	0x0000_0000	60.3.1.6/2600
PMC_BASE + 0x003C	FCCU Event Enable VD3 (FEE_VD3)	R/W	0x0000_00C1	60.3.1.7/2601
PMC_BASE + 0x0040	LVD108 Event Pending register VD4 (EPR_VD4)	R/W	0x0000_0000	60.3.1.8/2602
PMC_BASE + 0x0044	Reset Event Enable VD4 (REE_VD4)	R/W	0x0000_0001	60.3.1.9/2603
PMC_BASE + 0x0048	Reset Event Selection VD4 (RES_VD4)	R/W	0x0000_0000	60.3.1.10/2604
PMC_BASE + 0x004C	FCCU Event Enable VD4 (FEE_VD4)	R/W	0x0000_0001	60.3.1.11/2604
PMC_BASE + 0x0070	HVD140 Event Pending register VD7 (EPR_VD7)	R/W	0x0000_0000	60.3.1.12/2605
PMC_BASE + 0x0074	Reset Event Enable VD7 (REE_VD7)	R/W	0x0000_0041	60.3.1.13/2606
PMC_BASE + 0x0078	Reset Event Selection VD7 (RES_VD7)	R/W	0x0000_0000	60.3.1.14/2607
PMC_BASE + 0x007C	FCCU Event Enable VD7 (FEE_VD7)	R/W	0x0000_0041	60.3.1.15/2607
PMC_BASE + 0x0080	HVD145 Event Pending register VD8 (EPR_VD8)	R/W	0x0000_0000	60.3.1.16/2608
PMC_BASE + 0x0084	Reset Event Enable VD8 (REE_VD8)	R/W	0x0000_0001	60.3.1.17/2609
PMC_BASE + 0x0088	Reset Event Selection VD8 (RES_VD8)	R/W	0x0000_0000	60.3.1.18/2610
PMC_BASE + 0x008C	FCCU Event Enable VD8 (FEE_VD8)	R/W	0x0000_0001	60.3.1.19/2611
PMC_BASE + 0x0090	LVD270 Event Pending register VD9 (EPR_VD9)	R/W	0x0000_0000	60.3.1.20/2612
PMC_BASE + 0x0094	Reset Event Enable VD9 (REE_VD9)	R/W	0x0000_4741	60.3.1.21/2614
PMC_BASE + 0x0098	Reset Event Selection VD9 (RES_VD9)	R/W	0x0000_0000	60.3.1.22/2616
PMC_BASE + 0x009C	FCCU Event Enable VD9 (FEE_VD9)	R/W	0x0000_4341	60.3.1.23/2617
PMC_BASE + 0x00A0	LVD295 Event Pending register VD10 (EPR_VD10)	R/W	0x0000_0000	60.3.1.24/2619
PMC_BASE + 0x00A4	Reset Event Enable VD10 (REE_VD10)	R/W	0x0000_0040	60.3.1.25/2620

Table 1345. Power Management Controller Memory Map (continued)

Address	Register	Access	Reset Value	Section /Page
PMC_BASE + 0x00AC	FCCU Event Enable VD10 (FEE_VD10)	R/W	0x0000_0040	60.3.1.26/2620
PMC_BASE + 0x00C0	HVD360 Event Pending register VD12 (EPR_VD12)	R/W	0x0000_0000	60.3.1.27/2622
PMC_BASE + 0x00C4	Reset Event Enable VD12 (REE_VD12)	R/W	0x0000_0040	60.3.1.28/2622
PMC_BASE + 0x00C8	Reset Event Selection VD12 (RES_VD12)	R/W	0x0000_0000	60.3.1.29/2624
PMC_BASE + 0x00CC	FCCU Event Enable VD12 (FEE_VD12)	R/W	0x0000_0040	60.3.1.30/2625
PMC_BASE + 0x00D0	LVD360 Event Pending register VD13 (EPR_VD13)	R/W	0x0000_0000	60.3.1.31/2626
PMC_BASE + 0x00D4	Reset Event Enable VD13 (REE_VD13)	R/W	0x0000_8100	60.3.1.32/2626
PMC_BASE + 0x00D8	Reset Event Selection VD13 (RES_VD13)	R/W	0x0000_0000	60.3.1.33/2628
PMC_BASE + 0x00DC	FCCU Event Enable VD13 (FEE_VD13)	R/W	0x0000_8100	60.3.1.34/2628
PMC_BASE + 0x00E0	LVD400 Event Pending register VD14 (EPR_VD14)	R/W	0x0000_0000	60.3.1.35/2630
PMC_BASE + 0x00E4	Reset Event Enable VD14 (REE_VD14)	R/W	0x0000_8100	60.3.1.36/2631
PMC_BASE + 0x00E8	Reset Event Selection VD14 (RES_VD14)	R/W	0x0000_0000	60.3.1.37/2632
PMC_BASE + 0x00EC	FCCU Event Enable VD14 (FEE_VD14)	R/W	0x0000_8100	60.3.1.38/2633
PMC_BASE + 0x00F0	HVD600 Event Pending register VD15 (EPR_VD15)	R/W	0x0000_0000	60.3.1.39/2634
PMC_BASE + 0x00F4	Reset Event Enable VD15 (REE_VD15)	R/W	0x0000_8001	60.3.1.40/2635
PMC_BASE + 0x00F8	Reset Event Selection VD15 (RES_VD15)	R/W	0x0000_0000	60.3.1.41/2636
PMC_BASE + 0x00FC	FCCU Event Enable VD15 (FEE_VD15)	R/W	0x0000_8001	60.3.1.42/2638
PMC_BASE + 0x0104	Voltage Selection for I/O Segment (VSIO)	R/W	0x0000_0F00	60.3.1.43/2639
PMC_BASE + 0x0300	Temperature Event Pending register (EPR_TD)	R/W	0x0000_0000	60.3.1.44/2640
PMC_BASE + 0x0304	Temperature Reset Event Enable register (REE_TD)	R/W	0x0000_0000	60.3.1.45/2641
PMC_BASE + 0x0308	Temperature Reset Event Selection register (RES_TD)	R/W	0x0000_0000	60.3.1.46/2642
PMC_BASE + 0x030C	Temperature detector configuration register (CTL_TD)	R/W	0x0000_0003	60.3.1.47/2643
PMC_BASE + 0x0318	Temperature FCCU Event Enable register (FEE_TD)	R/W	0x0000_000D	60.3.1.48/2645

Table 1345. Power Management Controller Memory Map (continued)

Address	Register	Access	Reset Value	Section /Page
PMC_BASE + 0x0340	Voltage Detect User Mode Test Register (VD_UTST)	R/W	0x0000_0000	60.3.1.49/2646
PMC_BASE + 0x0344	ADC channel select register (ADC_CS)	R/W	0x0000_0000	60.3.1.50/2646
PMC_BASE + 0x034C	Voltage Regulator 1.2V Control Register (VREG1P2_CTRL)	R/W	0x0000_0000	60.3.1.51/2647
PMC_BASE + 0x0350	Module Control Register (MCR)	R/W	0x0000_0000	60.3.1.52/2647

60.3.1 Register descriptions

60.3.1.1 Supply Gauge Status Register (GR_S)

This status register contains the gauge of the indicated supply. This indicates the present state of the voltage detect events for each of the supplies. Each of the voltage detect signals for a given voltage are combined to form a single bit.

Figure 1437. Supply Gauge Register (GR_S)

Offset: PMC_BASE + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VD15	VD14	VD13	VD12	0	VD10	VD9	VD8	VD7	0	0	VD4	VD3	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1346. GR_S field descriptions

Field	Description
0–15	Reserved.
16 VD15	VD15 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V supplies. It is asserted when any of the supplies rise above the corresponding HVD threshold, and clears when all the supplies fall below the corresponding HVD threshold. All the 6.00 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 6.00 V supply.

Table 1346. GR_S field descriptions (continued)

Field	Description
17 VD14	VD14 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V supplies. It is asserted when any of the supplies fall below the corresponding LVD threshold, and clears when all the supplies rise above the corresponding LVD threshold. All the 4.00 V supplies are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 4.00 V supplies.
18 VD13	VD13 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3.6 V supplies. It is asserted when any of the supplies fall below the corresponding LVD threshold, and clears when all the supplies rise above the corresponding LVD threshold. All the 3.6 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage 3.6V supply.
19 VD12	VD12 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 3.60 V supply. It is asserted when the supply rises above the corresponding HVD threshold, and clears when the supply falls below the corresponding HVD threshold. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 3.60 V supply.
20	Reserved.
21 VD10	VD10 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 3.00 V supply.
22 VD9	VD9 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V supplies. It is asserted when any of the supplies fall below the corresponding LVD threshold, and clears when all the supplies rise above the corresponding LVD threshold. All of the 2.8 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.8 V supplies.
23	VD8 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 1.45 V supplies. It is asserted when any of the supplies rise above the corresponding HVD threshold, and clears when all the supplies fall below the corresponding HVD threshold. All the 1.45 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage 1.45 V supplies.
24 VD7	VD7 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 1.40 V supplies. It is asserted when any of the supplies rise above the corresponding HVD threshold, and clears when all the supplies fall below the corresponding HVD threshold. All the 1.40 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage 1.40 V supplies.
25-26	Reserved.

Table 1346. GR_S field descriptions (continued)

Field	Description
27 VD4	VD4 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detects for the low voltage 1.14 V supply associated with the PLL, Flash, and hot point. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage 1.14 V point supply.
28 VD3	VD3 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V supply associated with the core logic. It is asserted when any of the VD3 supplies fall below the corresponding LVD threshold, and clears when the supplies rise above the corresponding LVD threshold. All the 1.14 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on any of the low voltage 1.14 V supplies.
29-31	Reserved.

60.3.1.2 Pending Gauge Status Register (GR_P)

This configuration register contains a mirror of the contents of the Event Pending Registers, but in a format that includes one bit for each voltage detect signal. These bits indicate the state of the voltage detect events including whether a voltage level event has occurred at any time in the past but has not been cleared (via the EPR register). These bits are read only, and they are cleared by writing a one to the corresponding EPR register.

The Pending registers are only set on an active edge of an event, so a transition must be seen before they will become set.

The raw analog block voltage detect signals are also synchronized to the bus clock before being used for the pending registers.

Figure 1438. Pending Gauge Status Register (GR_P)

Offset: PMC_BASE + 0x0004

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	VD15_A	VD15_C	0	VD14_A	VD14_IM	0	0	VD13_IM	VD12_F	VD10_A
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VD10_F	VD9_O	0	VD9_IF	VD9_IJ	VD9_IM	VD9_F	0	VD9_C	VD8_C	VD8_F	VD7_C	VD4_C	VD3_P	VD3_F	VD3_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1347. GR_P field descriptions

Field	Description
0-5	Reserved.
6 VD15_A	VD15_A high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V ADC supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD15[HVD15_A] is cleared. 0 No occurrence has happened, or EPR_VD15[HVD15_A] has been cleared. 1 HVD occurrence detected on the high voltage ADC supply.
7 VD15_C	VD15_C high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD15[HVD15_C] is cleared. 0 No occurrence has happened, or EPR_VD15[HVD15_C] has been cleared. 1 HVD occurrence detected on the high voltage cold point supply.
8	Reserved.
9 VD14_A	VD14_A low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V ADC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD14[LVD14_A] is cleared. 0 No occurrence has happened, or EPR_VD14[LVD14_A] has been cleared. 1 LVD occurrence detected on the high voltage ADC supply.
10 VD14_IM	VD14_IM low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD14[LVD14_IM] is cleared. 0 No occurrence has happened, or EPR_VD14[LVD14_IM] has been cleared. 1 LVD occurrence detected on the high voltage I/O main supply.
11-12	Reserved.
13 VD13_IM	VD13_IM low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3.6 V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD13[LVD13_IM] is cleared. 0 No occurrence has happened, or EPR_VD13[LVD13_IM] has been cleared. 1 LVD occurrence detected on the high voltage I/O main supply.
14 VD12_F	VD12_F high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 3.6 V flash supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD12[HVD12_F] is cleared. 0 No occurrence has happened, or EPR_VD12[HVD12_F] has been cleared. 1 HVD occurrence detected on the high voltage flash supply.
15 VD10_A	VD10_A low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V ADC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD10[LVD10_A] is cleared. 0 No occurrence has happened, or EPR_VD10[LVD10_A] has been cleared. 1 LVD occurrence detected on the high voltage ADC supply.
16 VD10_F	VD10_F low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD10[LVD10_F] is cleared. 0 No occurrence has happened, or EPR_VD10[LVD10_F] has been cleared. 1 LVD occurrence detected on the high voltage flash supply.

Table 1347. GR_P field descriptions (continued)

Field	Description
17 VD9_O	VD9_O low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V OSC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_O] is cleared. 0 No occurrence has happened, or EPR_VD9[LVD9_O] has been cleared. 1 LVD occurrence detected on the high voltage OSC supply.
18	Reserved.
19 VD9_IF	VD9_IF low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O Flexray supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_IF] is cleared. 0 No occurrence has happened, or EPR_VD9[LVD9_IF] has been cleared. 1 LVD occurrence detected on the high voltage I/O Flexray supply.
20 VD9_IJ	VD9_IJ low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O JTAG supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_IJ] is cleared. 0 No occurrence has happened, or EPR_VD9[LVD9_IJ] has been cleared. 1 LVD occurrence detected on the high voltage I/O JTAG supply.
21 VD9_IM	VD9_IM low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_IM] is cleared. 0 No occurrence has happened, or EPR_VD9[LVD9_IM] has been cleared. 1 LVD occurrence detected on the high voltage I/O main supply.
22 VD9_F	VD9_F low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_F] is cleared. 0 No occurrence has happened, or EPR_VD9[LVD9_F] has been cleared. 1 LVD occurrence detected on the high voltage flash supply.
23	Reserved.
24 VD9_C	VD9_C low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V cold point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_C] is cleared. 0 No occurrence has happened, or EPR_VD9[LVD9_C] has been cleared. 1 LVD occurrence detected on the high voltage cold point supply.
25 VD8_C	VD8_C high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.45 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD8[HVD8_C] is cleared. 0 No occurrence has happened, or EPR_VD8[HVD8_C] has been cleared. 1 HVD occurrence detected on the low voltage cold point supply.
26 VD8_F	VD8_F high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.45 V flash supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD8[HVD8_F] is cleared. 0 No occurrence has happened, or EPR_VD8[HVD8_F] has been cleared. 1 HVD occurrence detected on the low voltage flash supply.

Table 1347. GR_P field descriptions (continued)

Field	Description
27 VD7_C	VD7_C high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.4 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD7[HVD7_C] is cleared. 0 No occurrence has happened, or EPR_VD7[HVD7_C] has been cleared. 1 HVD occurrence detected on the low voltage cold point supply.
28 VD4_C	VD4_C low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V cold point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD4[LVD4_C] is cleared. 0 No occurrence has happened, or EPR_VD4[LVD4_C] has been cleared. 1 LVD occurrence detected on the low voltage cold point supply.
29 VD3_P	VD3_P low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V PLL supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD3[LVD3_P] is cleared. 0 No occurrence has happened, or EPR_VD3[LVD3_P] has been cleared. 1 LVD occurrence detected on the low voltage PLL supply.
30 VD3_F	VD3_F low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD3[LVD3_F] is cleared. 0 No occurrence has happened, or EPR_VD3[LVD3_F] has been cleared. 1 LVD occurrence detected on the low voltage flash supply.
31 VD3_C	VD3_C low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V hot point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD3[LVD3_C] is cleared. 0 No occurrence has happened, or EPR_VD3[LVD3_C] has been cleared. 1 LVD occurrence detected on the low voltage hot point supply.

60.3.1.3 Interrupt Enable Pending Register (IE_P)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Event Pending Registers (EPR_XX), but in a format similar to the Pending Gauge Status register (GR_P). These bits indicate whether an interrupt occurs when the voltage event is seen. A '0' indicates that no interrupt is to be sent, and a '1' indicates that an interrupt is to occur when the voltage detect event occurs. These bits are enabled via a write of one to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

Figure 1439. Interrupt Enable Pending Register (IE_P)

Offset: PMC_BASE + 0x0008

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IE_EN	0	0	0	0	0	VD15IE_A	VD15IE_C	0	VD14IE_A	VD14IE_IM	0	0	VD13IE_IM	VD12IE_F	VD10IE_A
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VD10IE_F	VD9IE_O	0	VD9IE_IF	VD9IE_IJ	VD9IE_IM	VD9IE_F	0	VD9IE_C	VD8IE_C	VD8IE_F	VD7IE_C	VD4IE_C	VD3IE_P	VD3IE_F	VD3IE_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1348. IE_P field descriptions

Field	Description
0 IE_EN	IE_EN. Interrupt Enable. This bit determines whether any of the Interrupt Enable bits can be written. 0 No interrupt enables can be written. 1 Any interrupt enable can be written.
1-5	Reserved.
6 VD15IE_A	VD15IE_A Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
7 VD15IE_C	VD15IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
8	Reserved.
9 VD14IE_A	VD14IE_A Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
10 VD14IE_IM	VD14IE_IM Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
11-12	Reserved.

Table 1348. IE_P field descriptions (continued)

Field	Description
13 VD13IE_IM	VD13IE_IM Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
14 VD12IE_F	VD12IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
15 VD10IE_A	VD10IE_A Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
16 VD10IE_F	VD10IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
17 VD9IE_O	VD9IE_O Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
18	Reserved.
19 VD9IE_IF	VD9IE_IF Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
20 VD9IE_IJ	VD9IE_IJ Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
21 VD9IE_IM	VD9IE_IM Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
22 VD9IE_F	VD9IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
23	Reserved.
24 VD9IE_C	VD9IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

Table 1348. IE_P field descriptions (continued)

Field	Description
25 VD8IE_C	VD8IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
26 VD8IE_F	VD8IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
27 VD7IE_C	VD7IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
28 VD4IE_C	VD4IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
29 VD3IE_P	VD3IE_P Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
30 VD3IE_F	VD3IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
31 VD3IE_C	VD3IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

60.3.1.4 Event Pending Register (EPR_VD3)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. To clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1440. Event Pending Register (EPR_VD3)

Offset: PMC_BASE + 0x0030

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LVD3_P	LVD3_F	0	0	0	0	0	LVD3_C
W									w1c	w1c						w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1349. EPR_VD3 field descriptions

Field	Description
0–23	Reserved.
24 LVD3_P	LVD3_P flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.08 V PLL supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD3IE_P bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD3[LVD3_P] is also asserted, a system reset is generated, which clears VD3IE_P and negate the interrupt request.</i> If REE_VD3[LVD3_P] is asserted and a destructive reset is selected (via RES_VD3[LVD3_P] being 0), then a destructive reset is generated. If RES_VD3[LVD3_P] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage PLL supply.
25 LVD3_F	LVD3_F flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.08 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD3IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD3[LVD3_F] is also asserted, a system reset is generated, which clears VD3IE_F and negate the interrupt request.</i> If REE_VD3[LVD3_F] is asserted and a destructive reset is selected (via RES_VD3[LVD3_F] being 0), then a destructive reset is generated. If RES_VD3[LVD3_F] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage flash supply.

Table 1349. EPR_VD3 field descriptions (continued)

Field	Description
26–30	Reserved.
31 LVD3_C	LVD3_C flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.08 V hot point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD3IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD3[LVD3_C] is also asserted, a system reset is generated, <i>which clears VD3IE_C and negate the interrupt request.</i> If REE_VD3[LVD3_C] is asserted and a destructive reset is selected (via RES_VD3[LVD3_C] being 0), then a destructive reset is generated. If RES_VD3[LVD3_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage 1.14 V hot point supply.

60.3.1.5 Reset Event Enable Register (REE_VD3)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Figure 1441. Reset Event Enable Register (REE_VD3)

Offset: PMC_BASE + 0x0034

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LVD3_P	LVD3_F	0	0	0	0	0	LVD3_C
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1

Table 1350. REE_VD3 field descriptions

Field	Description
0–23	Reserved.
24 LVD3_P	LVD3_P reset enable. This bit defines whether an LVD assertion on the supply of the PLL generates a system reset. The RES_VD3[LVD3_P] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the PLL does not cause system reset. 1 Enabled. LVD assertion on the supply of the PLL causes system reset.

Table 1350. REE_VD3 field descriptions (continued)

Field	Description
25 LVD3_F	LVD3_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a system reset. The RES_VD3[LVD3_F] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset. 1 Enabled. LVD assertion on the supply of the flash causes system reset.
26–30	Reserved.
31 LVD3_C	LVD3_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD3[LVD3_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

60.3.1.6 Reset Event Select Register (RES_VD3)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Figure 1442. Reset Event Select Register (RES_VD3)

Offset: PMC_BASE + 0x0038

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LVD3_P	LVD3_F	0	0	0	0	0	LVD3_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1351. RES_VD3 field descriptions

Field	Description
0–23	Reserved.
24 LVD3_P	LVD3_P reset event select. This bit defines whether an LVD assertion on the supply of the PLL generates a destructive reset or a functional reset. The REE_VD3[LVD3_P] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the PLL causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the PLL causes a functional system reset.

Table 1351. RES_VD3 field descriptions (continued)

Field	Description
25 LVD3_F	LVD3_F reset event select. This bit defines whether an LVD assertion on the supply of the flash generates a destructive reset or a functional reset. The REE_VD3[LVD3_F] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the flash causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the flash causes a functional system reset.
26–30	Reserved.
31 LVD3_C	LVD3_C reset event select. This bit defines whether an LVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD3[LVD3_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

60.3.1.7 FCCU Event Enable Register (FEE_VD3)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1443. FCCU Event Enable Register (FEE_VD3)

Offset: PMC_BASE + 0x003C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FEE3_P	FEE3_F	0	0	0	0	0	FEE3_C
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1

Table 1352. FEE_VD3 field descriptions

Field	Description
0–23	Reserved.
24 FEE3_P	FEE3_P FCCU event enable. This bit defines whether an LVD assertion on the supply of the PLL generates an FCCU event. 0 Disabled. LVD assertion on the supply of the PLL does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the PLL causes an FCCU event.
25 FEE3_F	FEE3_F FCCU event enable. This bit defines whether an LVD assertion on the supply of the flash generates an FCCU event. 0 Disabled. LVD assertion on the supply of the flash does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the flash causes an FCCU event.
26–30	Reserved.
31 FEE3_C	FEE3_C FCCU event enable. This bit defines whether an LVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. LVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the cold point causes an FCCU event.

60.3.1.8 Event Pending Register (EPR_VD4)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1444. Event Pending Register (EPR_VD4)

Offset: PMC_BASE + 0x0040

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LVD4_C
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1353. EPR_VD4 field descriptions

Field	Description
0–30	Reserved.
31 LVD4_C	LVD4_C flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V cold point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD4IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD4[LVD4_C] is also asserted, a system reset is generated, <i>which clears VD4IE_C and negate the interrupt request.</i> If REE_VD4[LVD4_C] is asserted and a destructive reset is selected (via RES_VD4[LVD4_C] being 0), then a destructive reset is generated. If RES_VD4[LVD4_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage cold point supply.

60.3.1.9 Reset Event Enable Register (REE_VD4)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Figure 1445. Reset Event Enable Register (REE_VD4)

Offset: PMC_BASE + 0x0044

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LVD4_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 1354. REE_VD4 field descriptions

Field	Description
0–30	Reserved.
31 LVD4_C	LVD4_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD4[LVD4_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

60.3.1.10 Reset Event Select Register (RES_VD4)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Figure 1446. Reset Event Select Register (RES_VD4)

Offset: PMC_BASE + 0x0048

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LVD4_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1355. RES_VD4 field descriptions

Field	Description
0–30	Reserved.
31 LVD4_C	LVD4_H reset event select. This bit defines whether an LVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD4[LVD4_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

60.3.1.11 FCCU Event Enable Register (FEE_VD4)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1447. FCCU Event Enable Register (*FEE_VD4*)

Offset: PMC_BASE + 0x004C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE4_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 1356. FEE_VD4 field descriptions

Field	Description
0–30	Reserved.
31 FEE4_C	FEE4_C FCCU event enable. This bit defines whether an LVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. LVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the cold point causes an FCCU event.

60.3.1.12 Event Pending Register (*EPR_VD7*)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1448. Event Pending Register (*EPR_VD7*)

Offset: PMC_BASE + 0x0070

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD7_C
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1357. EPR_VD7 field descriptions

Field	Description
0–30	Reserved.
31 HVD7_C	HVD7_C flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.4 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the supply falls below its corresponding HVD threshold and a one is written to this bit location. <i>If the VD7IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD7[HVD7_C] is also asserted, a system reset is generated, <i>which clears VD7IE_C and negate the interrupt request.</i> If REE_VD7[HVD7_C] is asserted and a destructive reset is selected (via RES_VD7[HVD7_C] being 0), then a destructive reset is generated. If RES_VD7[HVD7_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage cold point supply.

60.3.1.13 Reset Event Enable Register (REE_VD7)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Figure 1449. Reset Event Enable Register (REE_VD7)

Offset: PMC_BASE + 0x0074

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD7_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 1358. REE_VD7 field descriptions

Field	Description
0–30	Reserved.
31 HVD7_C	HVD7_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD7[HVD7_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

60.3.1.14 Reset Event Select Register (RES_VD7)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Figure 1450. Reset Event Select Register (RES_VD7)

Offset: PMC_BASE + 0x0078

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD7_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1359. RES_VD7 field descriptions

Field	Description
0–30	Reserved.
31 HVD7_C	HVD7_C reset event select. This bit defines whether an HVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD7[HVD7_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.

60.3.1.15 FCCU Event Enable Register (FEE_VD7)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1451. FCCU Event Enable Register (*FEE_VD7*)

Offset: PMC_BASE + 0x007C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE7_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 1360. FEE_VD7 field descriptions

Field	Description
0–30	Reserved.
31 FEE7_C	FEE7_C FCCU event enable. This bit defines whether an HVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. HVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the cold point causes an FCCU event.

60.3.1.16 Event Pending Register (*EPR_VD8*)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1452. Event Pending Register (*EPR_VD8*)

Offset: PMC_BASE + 0x0080

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	HVD8_F	0	0	0	0	0	HVD8_C
W										w1c						w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1361. EPR_VD8 field descriptions

Field	Description
0–24	Reserved.
25 HVD8_F	HVD8_F flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.45 V Flash supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the supply falls below its corresponding HVD threshold and a one is written to this bit location. <i>If the VD8IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD8[HVD8_F] is also asserted, a system reset is generated, <i>which clears VD8IE_F and negate the interrupt request.</i> If REE_VD8[HVD8_F] is asserted and a destructive reset is selected (via RES_VD8[HVD8_F] being 0), then a destructive reset is generated. If RES_VD8[HVD8_F] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage Flash supply.
26–30	Reserved.
31 HVD8_C	HVD8_C flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.45 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the supply falls below its corresponding HVD threshold and a one is written to this bit location. <i>If the VD8IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD8[HVD8_C] is also asserted, a system reset is generated, <i>which clears VD8IE_C and negate the interrupt request.</i> If REE_VD8[HVD8_C] is asserted and a destructive reset is selected (via RES_VD8[HVD8_C] being 0), then a destructive reset is generated. If RES_VD8[HVD8_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage cold point supply.

60.3.1.17 Reset Event Enable Register (REE_VD8)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) unless in test mode.

Figure 1453. Reset Event Enable Register (REE_VD8)

Offset: PMC_BASE + 0x0084

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	HVD8_F	0	0	0	0	0	HVD8_C
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Table 1362. REE_VD8 field descriptions

Field	Description
0–24	Reserved.
25 HVD8_F	HVD8_F reset enable. This bit defines whether an HVD assertion on the supply of the Flash generates a system reset. The RES_VD8[HVD8_F] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the Flash does not cause system reset. 1 Enabled. HVD assertion on the supply of the Flash causes system reset.
26–30	Reserved.
31 HVD8_C	HVD8_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD8[HVD8_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

60.3.1.18 Reset Event Select Register (RES_VD8)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Figure 1454. Reset Event Select Register (RES_VD8)

Offset: PMC_BASE + 0x0088

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	HVD8_F	0	0	0	0	0	HVD8_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1363. RES_VD8 field descriptions

Field	Description
0–24	Reserved.
25 HVD8_F	HVD8_F reset event select. This bit defines whether an HVD assertion on the supply of the Flash generates a destructive reset or a functional reset. The REE_VD8[HVD8_F] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the Flash causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the Flash causes a functional system reset.
26–30	Reserved.
31 HVD8_C	HVD8_C reset event select. This bit defines whether an HVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD8[HVD8_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.

60.3.1.19 FCCU Event Enable Register (FEE_VD8)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1455. FCCU Event Enable Register (FEE_VD8)

Offset: PMC_BASE + 0x008C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	FEE8_F	0	0	0	0	0	FEE8_C
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Table 1364. FEE_VD8 field descriptions

Field	Description
0–24	Reserved.
25 FEE8_F	FEE8_F FCCU event enable. This bit defines whether an HVD assertion on the supply of the Flash generates an FCCU event. 0 Disabled. HVD assertion on the supply of the Flash does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the Flash causes an FCCU event.
26–30	Reserved.
31 FEE8_C	FEE8_C FCCU event enable. This bit defines whether an HVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. HVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the cold point causes an FCCU event.

60.3.1.20 Event Pending Register (EPR_VD9)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1456. Event Pending Register (EPR_VD9)

Offset: PMC_BASE + 0x0090

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LVD9_O	0	0	0	LVD9_IF	LVD9_IJ	LVD9_IM	0	LVD9_F	0	0	0	0	0	LVD9_C
W		w1c				w1c	w1c	w1c		w1c						w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1365. EPR_VD9 field descriptions

Field	Description
0–16	Reserved.
17 LVD9_O	LVD9_O flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V OSC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD9IE_O bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVDn_O] is also asserted, a system reset is generated, which clears VD9IE_O and negate the interrupt request.</i> If REE_VD9[LVD9_O] is asserted a POR reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage OSC supply.
18–20	Reserved.
21 LVD9_IF	LVD9_IF flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O Flexray supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD9IE_IF bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVD9_IF] is also asserted, a system reset is generated, which clears VD9IE_IF and negate the interrupt request.</i> If REE_VD9[LVD9_IF] is asserted and a destructive reset is selected (via RES_VD9[LVD9_IF] being 0), then a destructive reset is generated. If RES_VD9[LVD9_IF] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O Flexray supply.

Table 1365. EPR_VD9 field descriptions (continued)

Field	Description
22 LVD9_IJ	LVD9_IJ flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O JTAG supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD9IE_IJ bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD9[LVD9_IJ] is also asserted, a system reset is generated, <i>which clears VD9IE_IJ and negate the interrupt request.</i> If REE_VD9[LVD9_IJ] is asserted a POR reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O JTAG supply.
23 LVD9_IM	LVD9_IM flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD9IE_IM bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD9[LVD9_IM] is also asserted, a system reset is generated, <i>which clears VD9IE_IM and negate the interrupt request.</i> If REE_VD9[LVD9_A] is asserted a POR reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O main supply.
24	Reserved.
25 LVD9_F	LVD9_F flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD9IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD9[LVD9_F] is also asserted, a system reset is generated, <i>which clears VD9IE_F and negate the interrupt request.</i> If REE_VD9[LVD9_F] is asserted a POR reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage flash supply.
26–30	Reserved.
31 LVD9_C	LVD9_C flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V cold point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD9IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD9[LVD9_C] is also asserted, a system reset is generated, <i>which clears VD9IE_C and negate the interrupt request.</i> If REE_VD9[LVD9_C] is asserted a POR reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage cold point supply.

60.3.1.21 Reset Event Enable Register (REE_VD9)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event.

Figure 1457. Reset Event Enable Register (REE_VD9)

Offset: PMC_BASE + 0x0094

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LVD9_O	0	0	0	LVD9_IF	LVD9_IJ	LVD9_IM	0	LVD9_F	0	0	0	0	0	LVD9_C
W																
Reset	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	1

Table 1366. REE_VD9 field descriptions

Field	Description
0–16	Reserved.
17 LVD9_O	LVD9_O reset enable. This bit defines whether an LVD assertion on the supply of the OSC generates a POR reset. 0 Disabled. LVD assertion on the supply of the OSC does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the OSC causes a POR reset.
18–20	Reserved.
21 LVD9_IF	LVD9_IF reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the Flexray pins generates a system reset. The RES_VD9[LVD9_IF] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the Flexray pins does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the Flexray pins causes a POR reset.
22 LVD9_IJ	LVD9_IJ reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the JTAG pins generates a POR reset. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the JTAG pins does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the JTAG pins causes a POR reset.
23 LVD9_IM	LVD9_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates a POR reset. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes a POR reset.
24	Reserved.

Table 1366. REE_VD9 field descriptions (continued)

Field	Description
25 LVD9_F	LVD9_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a POR reset. 0 Disabled. LVD assertion on the supply of the flash does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the flash causes a POR reset.
26–30	Reserved.
31 LVD9_C	LVD9_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a POR reset. 0 Disabled. LVD assertion on the supply of the cold point does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the cold point causes a POR reset.

60.3.1.22 Reset Event Select Register (RES_VD9)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. All of the LVD9 signals are used to generate a POR except for LVD9_IF. The LVD9_IF is the only event at this voltage level that can cause a functional or destructive reset. All the others cause a POR (if enabled via the corresponding REE bit) or do not cause a reset.

Figure 1458. Reset Event Select Register (RES_VD9)

Offset: PMC_BASE + 0x0098

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	LVD9_IF	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1367. RES_VD9 field descriptions

Field	Description
0–20	Reserved.
21 LVD9_IF	LVD9_IF reset event select. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the Flexray pins generates a destructive reset or a functional reset. The REE_VD9[LVD9_IF] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the I/O segment that contains the Flexray pins causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the I/O segment that contains the Flexray pins causes a functional system reset.
22–31	Reserved.

60.3.1.23 FCCU Event Enable Register (FEE_VD9)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1459. FCCU Event Enable Register (FEE_VD9)

Offset: PMC_BASE + 0x009C

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W		FEE9_O				FEE9_IF	FEE9_IJ	FEE9_IM		FEE9_F						FEE9_C
Reset	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	1

Table 1368. FEE_VD9 field descriptions

Field	Description
0–16	Reserved.
17 FEE9_O	FEE9_O FCCU event enable. This bit defines whether an LVD assertion on the supply of the OSC generates an FCCU event. 0 Disabled. LVD assertion on the supply of the OSC does not cause system reset. 1 Enabled. LVD assertion on the supply of the OSC causes system reset.
18–20	Reserved.

Table 1368. FEE_VD9 field descriptions (continued)

Field	Description
21 FEE9_IF	FEE9_IF FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the Flexray pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the Flexray pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the Flexray pins causes an FCCU event.
22 FEE9_IJ	FEE9_IJ FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the JTAG pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the JTAG pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the JTAG pins causes FCCU event.
23 FEE9_IM	FEE9_IM FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes an FCCU event.
24	Reserved.
25 FEE9_F	FEE9_F FCCU event enable. This bit defines whether an LVD assertion on the supply of the flash generates an FCCU event 0 Disabled. LVD assertion on the supply of the flash does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the flash causes an FCCU event.
26–30	Reserved.
31 FEE9_C	FEE9_C FCCU event enable. This bit defines whether an LVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. LVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the cold point causes an FCCU event.

60.3.1.24 Event Pending Register (EPR_VD10)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1460. Event Pending Register (EPR_VD10)

Offset: PMC_BASE + 0x00A0

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVD10_A	0	0	0	0	0	0	0	0	LVD10_F	0	0	0	0	0	0
W	w1c									w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1369. EPR_VD10 field descriptions

Field	Description
0–17	Reserved.
16 LVD10_A	LVD10_A flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V ADC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD10IE_A bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD10[LVD10_A] is also asserted, a POR reset is generated, <i>which clears VD10IE_A and negate the interrupt request.</i> If REE_VD10[LVD10_A] is asserted a POR reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage ADC supply.
15–24	Reserved.
25 LVD10_F	LVD10_F flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD10IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD10[LVD10_F] is also asserted, a POR reset is generated, <i>which clears VD10IE_F and negate the interrupt request.</i> If REE_VD10[LVD10_F] is asserted a POR reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage flash supply.
26–31	Reserved.

60.3.1.25 Reset Event Enable Register (REE_VD10)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Figure 1461. Reset Event Enable Register (REE_VD10)

Offset: PMC_BASE + 0x00A4

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVD10_A	0	0	0	0	0	0	0	0	LVD10_F	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 1370. REE_VD10 field descriptions

Field	Description
0–15	Reserved.
16 LVD10_A	LVD10_A reset enable. This bit defines whether an LVD assertion on the supply of the ADC generates a system reset. 0 Disabled. LVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the supply of the ADC causes system reset.
17–24	Reserved.
25 LVD10_F	LVD10_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a system reset. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset. 1 Enabled. LVD assertion on the supply of the flash causes system reset.
26–31	Reserved.

60.3.1.26 FCCU Event Enable Register (FEE_VD10)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag

bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1462. FCCU Event Enable Register (FEE_VD10)

Offset: PMC_BASE + 0x00AC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FEE10_A	0	0	0	0	0	0	0	0	FEE10_F	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 1371. FEE_VD10 field descriptions

Field	Description
0–15	Reserved.
16 FEE10_A	FEE10_A FCCU event enable. This bit defines whether an LVD assertion on the supply of the ADC generates an FCCU event. 0 Disabled. LVD assertion on the supply of the ADC does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the ADC causes an FCCU event.
17–24	Reserved.
25 FEE10_F	FEE10_F FCCU event enable. This bit defines whether an LVD assertion on the supply of the flash generates an FCCU event. 0 Disabled. LVD assertion on the supply of the flash does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the flash causes an FCCU event.
26–31	Reserved.

60.3.1.27 Event Pending Register (EPR_VD12)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1463. Event Pending Register (EPR_VD12)

Offset: PMC_BASE + 0x00C0

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	HVD12_F	0	0	0	0	0	0
W										w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1372. EPR_VD12 field descriptions

Field	Description
0–24	Reserved.
25 HVD12_F	HVD12_F flag. This bit is the high-voltage status flag associated with the voltage level detect for the high voltage 3.6 V flash supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the supply falls below its corresponding HVD threshold and a one is written to this bit location. <i>If the VD12IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD12[HVD12_F] is also asserted, a system reset is generated, which clears VD12IE_F and negate the interrupt request.</i> If REE_VD12[HVD12_F] is asserted and a destructive reset is selected (via RES_VD12[HVD12_F] being 0), then a destructive reset is generated. If RES_VD12[HVD12_F] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage flash supply.
26–31	Reserved.

60.3.1.28 Reset Event Enable Register (REE_VD12)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from

the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Figure 1464. Reset Event Enable Register (REE_VD12)

Offset: PMC_BASE + 0x00C4

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	HVD12_F	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 1373. REE_VD12 field descriptions

Field	Description
0–24	Reserved.
25 HVD12_F	HVD12_F reset enable. This bit defines whether an HVD assertion on the supply of the flash generates a system reset. The RES_VD12[HVD12_F] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the flash does not cause system reset. 1 Enabled. HVD assertion on the supply of the flash causes system reset.
26–31	Reserved.

60.3.1.29 Reset Event Select Register (RES_VD12)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Figure 1465. Reset Event Select Register (RES_VD12)

Offset: PMC_BASE + 0x00C8

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	HVD12_F	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1374. RES_VD12 field descriptions

Field	Description
0–24	Reserved.
25 HVD12_F	HVD12_F reset event select. This bit defines whether an HVD assertion on the supply of the flash generates a destructive reset or a functional reset. The REE_VD12[HVD12_F] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the flash causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the flash causes a functional system reset.
26–31	Reserved.

60.3.1.30 FCCU Event Enable Register (FEE_VD12)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1466. FCCU Event Enable Register (FEE_VD12)

Offset: PMC_BASE + 0x00CC

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	FEE12_F	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 1375. FEE_VD12 field descriptions

Field	Description
0–24	Reserved.
25 FEE12_F	FEE12_F FCCU event enable. This bit defines whether an HVD assertion on the supply of the flash generates an FCCU event. 0 Disabled. HVD assertion on the supply of the flash does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the flash causes an FCCU event.
26–31	Reserved.

60.3.1.31 Event Pending Register (EPR_VD13)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1467. Event Pending Register (EPR_VD13)

Offset: PMC_BASE + 0x00D0												Access: User read/write				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	LVD13_IM	0	0	0	0	0	0	0	0
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1376. EPR_VD13 field descriptions

Field	Description
0–22	Reserved.
23 LVD13_IM	LVD13_IM flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3.6 V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD13IE_IM bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD13[LVD13_IM] is also asserted, a system reset is generated, <i>which clears VD13IE_IM and negate the interrupt request.</i> If REE_VD13[LVD13_IM] is asserted and a destructive reset is selected (via RES_VD13[LVD13_IM] being 0), then a destructive reset is generated. If RES_VD13[LVD13_IM] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O main supply.
24–31	Reserved.

60.3.1.32 Reset Event Enable Register (REE_VD13)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from

the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Figure 1468. Reset Event Enable Register (REE_VD13)

Offset: PMC_BASE + 0x00D4

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	LVD13_IM	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 1377. REE_VD13 field descriptions

Field	Description
0–22	Reserved.
23 LVD13_IM	LVD13_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates a system reset. The RES_VD13[LVD13_IM] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes system reset.
24–31	Reserved.

60.3.1.33 Reset Event Select Register (RES_VD13)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Figure 1469. Reset Event Select Register (RES_VD13)

Offset: PMC_BASE + 0x00D8

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W								LVD13_IM								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1378. RES_VD13 field descriptions

Field	Description
0–22	Reserved.
23 LVD13_IM	LVD13_IM reset event select. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates a destructive reset or a functional reset. The REE_VD13[LVD13_IM] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the I/O segment that contains the main pins causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the I.O segment that contains the main pins causes a functional system reset.
24–31	Reserved.

60.3.1.34 FCCU Event Enable Register (FEE_VD13)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag

bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1470. FCCU Event Enable Register (FEE_VD13)

Offset: PMC_BASE + 0x00DC

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	FEE13_IM	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 1379. FEE_VD13 field descriptions

Field	Description
0–22	Reserved.
23 FEE13_IM	FEE13_IM FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes an FCCU event.
24–31	Reserved.

60.3.1.35 Event Pending Register (EPR_VD14)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1471. Event Pending Register (EPR_VD14)

Offset: PMC_BASE + 0x00E0												Access: User read/write			
31 30 29 28				27 26 25 24				23 22 21 20				19 18 17 16			
0 1 2 3				4 5 6 7				8 9 10 11				12 13 14 15			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15 14 13 12				11 10 9 8				7 6 5 4				3 2 1 0			
16 17 18 19				20 21 22 23				24 25 26 27				28 29 30 31			
R	LVD14_A	0	0	0	0	0	0	LVD14_IM	0	0	0	0	0	0	0
W	w1c							w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1380. EPR_VD14 field descriptions

Field	Description
0–15	Reserved.
16 LVD14_A	LVD14_A flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V ADC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD14IE_A bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD14[LVD14_A] is also asserted, a system reset is generated, <i>which clears VD14IE_A and negate the interrupt request.</i> If REE_VD14[LVD14_A] is asserted and a destructive reset is selected (via RES_VD14[LVD14_A] being 0), then a destructive reset is generated. If RES_VD14[LVD14_A] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage ADC supply.
17–22	Reserved.

Table 1380. EPR_VD14 field descriptions (continued)

Field	Description
23 LVD14_IM	LVD14_IM flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V I/O Main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. <i>If the VD14IE_IM bit is also asserted, a low-voltage interrupt is sent to the CPU.</i> If REE_VD14[LVD14_IM] is also asserted, a system reset is generated, <i>which clears VD14IE_IM and negate the interrupt request.</i> If REE_VD14[LVD14_IM] is asserted and a destructive reset is selected (via RES_VD14[LVD14_IM] being 0), then a destructive reset is generated. If RES_VD14[LVD14_IM] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O Main supply.
24–31	Reserved.

60.3.1.36 Reset Event Enable Register (REE_VD14)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Figure 1472. Reset Event Enable Register (REE_VD14)

Offset: PMC_BASE + 0x00E4

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVD14_A	0	0	0	0	0	0	LVD14_IM	0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 1381. REE_VD14 field descriptions

Field	Description
0–15	Reserved.
16 LVD14_A	LVD14_A reset enable. This bit defines whether an LVD assertion on the supply of the ADC generates a system reset. The RES_VD14[LVD14_A] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the supply of the ADC causes system reset.
17–22	Reserved.
23 LVD14_IM	LVD14_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O Main generates a system reset. The RES_VD14[LVD14_IM] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O Main does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O Main causes system reset.
24–31	Reserved.

60.3.1.37 Reset Event Select Register (RES_VD14)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Figure 1473. Reset Event Select Register (RES_VD14)

Offset: PMC_BASE + 0x00E8

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVD14_A	0	0	0	0	0	0	LVD14_IM	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1382. RES_VD14 field descriptions

Field	Description
0–15	Reserved.
16 LVD14_A	LVD14_A reset event select. This bit defines whether an LVD assertion on the supply of the ADC generates a destructive reset or a functional reset. The REE_VD14[LVD14_A] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the ADC causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the ADC causes a functional system reset.
17–22	Reserved.
23 LVD14_IM	LVD14_IM reset event select. This bit defines whether an LVD assertion on the supply of the IO Main generates a destructive reset or a functional reset. The REE_VD14[LVD14_IM] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the I/O Main causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the I/O Main causes a functional system reset.
24–31	Reserved.

60.3.1.38 FCCU Event Enable Register (FEE_VD14)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1474. FCCU Event Enable Register (FEE_VD14)

Offset: PMC_BASE + 0x00EC

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FEE14_A	0	0	0	0	0	0	0	FEE14_IM	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 1383. FEE_VD14 field descriptions

Field	Description
0–15	Reserved.
16 FEE14_A	FEE14_A FCCU event enable. This bit defines whether an LVD assertion on the supply of the ADC generates an FCCU event. 0 Disabled. LVD assertion on the supply of the ADC does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the ADC causes an FCCU event.
17–22	Reserved.
23 FEE14_IM	FEE14_IM FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O Main generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O Main does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O Main causes an FCCU event.
24–31	Reserved.

60.3.1.39 Event Pending Register (EPR_VD15)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1475. Event Pending Register (EPR_VD15)

Offset: PMC_BASE + 0x00F0

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HVD15_A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD15_C
W	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1384. EPR_VD15 field descriptions

Field	Description
0–15	Reserved.
16 HVD15_A	HVD15_A flag. This bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V ADC supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the supply falls below its corresponding HVD threshold and a one is written to this bit location. <i>If the VD15IE_A bit is also asserted, a high-voltage interrupt is sent to the CPU.</i> If REE_VD15[HVD15_A] is also asserted, a system reset is generated, <i>which clears VD15IE_A and negate the interrupt request.</i> If REE_VD15[HVD15_A] is asserted and a destructive reset is selected (via RES_VD15[HVD15_A] being 0), then a destructive reset is generated. If RES_VD15[HVD15_A] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage ADC supply.
17–30	Reserved.
31 HVD15_C	HVD15_C flag. This bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the supply falls below its corresponding HVD threshold and a one is written to this bit location. <i>If the VD15IE_C bit is also asserted, a high-voltage interrupt is sent to the CPU.</i> If REE_VD15[HVD15_C] is also asserted, a system reset is generated, <i>which clears VD15IE_C and negate the interrupt request.</i> If REE_VD15[HVD15_C] is asserted and a destructive reset is selected (via RES_VD15[HVD15_C] being 0), then a destructive reset is generated. If RES_VD15[HVD15_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage cold point supply.

60.3.1.40 Reset Event Enable Register (REE_VD15)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Figure 1476. Reset Event Enable Register (REE_VD15)

Offset: PMC_BASE + 0x00F4

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HVD15_A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD15_C
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 1385. REE_VD15 field descriptions

Field	Description
0–15	Reserved.
16 HVD15_A	HVD15_A reset enable. This bit defines whether an HVD assertion on the supply of the ADC generates a system reset. The RES_VD15[HVD15_A] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. HVD assertion on the supply of the ADC causes system reset.
17–30	Reserved.
31 HVD15_C	HVD15_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD15[HVD15_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

60.3.1.41 Reset Event Select Register (RES_VD15)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Figure 1477. Reset Event Select Register (*RES_VD15*)

Offset: PMC_BASE + 0x00F8

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HVD15_A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD15_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1386. RES_VD15 field descriptions

Field	Description
0–15	Reserved.
16 HVD15_A	HVD15_A reset event select. This bit defines whether an HVD assertion on the supply of the ADC generates a destructive reset or a functional reset. The REE_VD15[HVD15_A] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the ADC causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the ADC causes a functional system reset.
17–30	Reserved.
31 HVD15_C	HVD15_C reset event select. This bit defines whether an HVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD15[HVD15_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.

60.3.1.42 FCCU Event Enable Register (FEE_VD15)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Figure 1478. FCCU Event Enable Register (FEE_VD15)

Offset: PMC_BASE + 0x00FC

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FEE15_A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE15_C
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 1387. FEE_VD15 field descriptions

Field	Description
0–15	Reserved.
16 FEE15_A	FEE15_A FCCU event enable. This bit defines whether an HVD assertion on the supply of the ADC generates an FCCU event. 0 Disabled. HVD assertion on the supply of the ADC does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the ADC causes an FCCU event.
17–30	Reserved.
31 FEE15_C	FEE15_C FCCU event enable. This bit defines whether an HVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. HVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the cold point causes an FCCU event.

60.3.1.43 Voltage Supply for I/O Segments Register (VSIO)

The Voltage Supply for I/O Segments (VSIO) register indicates if the I/O segment indicated uses a 3.3V or 5.0V supply. These bits all reset to use a 5.0V supply.

Figure 1479. Voltage Supply for I/O Segments Register (VSIO)

Offset: PMC_BASE + 0x0104

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	VSIO_IF	VSIO_IJ	VSIO_IM	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Table 1388. VSIO field descriptions

Field	Description
0–20	Reserved.
21 VSIO_IF	VSIO_IF control. This bit is used to control the supply levels used by the Flexray I/O segment. 0 The I/O segment uses a 3.3V supply. 1 The I/O segment uses a 5.0V supply.
22 VSIO_IJ	VSIO_IJ control. This bit is used to control the supply levels used by the JTAG I/O segment. 0 The I/O segment uses a 3.3V supply. 1 The I/O segment uses a 5.0V supply.
23 VSIO_IM	VSIO_IM control. This bit is used to control the supply levels used by the Main I/O segment. 0 The I/O segment uses a 3.3V supply. 1 The I/O segment uses a 5.0V supply.
24–31	Reserved.

60.3.1.44 Event Pending Register (EPR_TD)

This Event Pending Register indicates the present and past state of the temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Figure 1480. Event Pending Register (EPR_TD)

Offset: PMC_BASE + 0x0300

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_3	TEMP_2	0	TEMP_0
W													w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1389. EPR_TD field descriptions

Field	Description
0–27	Reserved.
28 TEMP_3	TEMP_3 flag. This bit is the temperature status flag associated with the temperature for the <i>high</i> temperature sensor point (165C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a one is written to this bit location. <i>If the IETD_3 bit is also asserted, an interrupt is sent to the CPU.</i> If REE_TD[TEMP_3] is also asserted, a system reset is generated, <i>which clears IETD_3 and negate the interrupt request.</i> If REE_TD[TEMP_3] is asserted and a destructive reset is selected (via RES_TD[TEMP_3] being 0), then a destructive reset is generated. If RES_TD[TEMP_3] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
29 TEMP_2	TEMP_2 flag. This bit is the temperature status flag associated with the temperature for the temperature sensor point (150C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a one is written to this bit location. <i>If the IETD_2 bit is also asserted, an interrupt is sent to the CPU.</i> If REE_TD[TEMP_2] is also asserted, a system reset is generated, <i>which clears IETD_2 and negate the interrupt request.</i> If REE_TD[TEMP_2] is asserted and a destructive reset is selected (via RES_TD[TEMP_2] being 0), then a destructive reset is generated. If RES_TD[TEMP_2] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.

Table 1389. EPR_TD field descriptions (continued)

Field	Description
30	Reserved.
31 TEMP_0	TEMP_0 flag. This bit is the temperature status flag associated with the temperature for the <i>cold</i> temperature sensor point. It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a one is written to this bit location. <i>If the IETD_0 bit is also asserted, an interrupt is sent to the CPU.</i> If REE_TD[TEMP_0] is also asserted, a system reset is generated, <i>which clears IETD_0 and negate the interrupt request.</i> If REE_TD[TEMP_0] is asserted and a destructive reset is selected (via RES_TD[TEMP_0] being 0), then a destructive reset is generated. If RES_TD[TEMP_0] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.

60.3.1.45 Reset Event Enable Register (REE_TD)

This Reset Event Enable Register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Figure 1481. Reset Event Enable Register (REE_TD)

Offset: PMC_BASE + 0x0304

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_3	TEMP_2	0	TEMP_0
W													TEMP_3	TEMP_2		TEMP_0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1390. REE_TD field descriptions

Field	Description
0–27	Reserved.
28 TEMP_3	TEMP_3 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_3] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.

Table 1390. REE_TD field descriptions (continued)

Field	Description
29 TEMP_2	TEMP_2 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
30	Reserved.
31 TEMP_0	TEMP_0 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.

60.3.1.46 Reset Event Select Register (RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the temperature sensor event causes either a destructive or functional reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Figure 1482. Reset Event Select Register (RES_TD)

Offset: PMC_BASE + 0x0308

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_3	TEMP_2	0	TEMP_1
W													TEMP_3	TEMP_2		TEMP_1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1391. RES_TD field descriptions

Field	Description
0–27	Reserved.
28 TEMP_3	TEMP_3 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_3] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.

Table 1391. RES_TD field descriptions (continued)

Field	Description
29 TEMP_2	TEMP_2 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_2] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
30	Reserved.
31 TEMP_0	TEMP_0 reset event select. This bit defines whether an temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_0] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.

60.3.1.47 Temperature Sensor Configuration Register (CTL_TD)

The Temperature Sensor Registers (CTL_TD) contains two enables that must both be enabled in order for the Temperature Sensor to be operable. It also includes two four bit signed trim adjustment register bits that are used to modify both TS trim values (one bus for over temp and one bus for under temp). Finally, this register includes Interrupt Enables (IE) for each of the Temp Sensor EPR bits.

The use case for the TRIM_ADJ_OVER and _UNDER bits would be to introduce additional hysteresis into the Temp Sensor events.

For instance, the TRIM_ADJ_OVER bits could be written to 1111 (-7) and the Temp Sensor event set to be an interrupt. When this interrupt occurs, the TRIM_ADJ_OVER bits could be updated to 0111 (+7), the interrupt cleared, and the Temp Sensor event set to be a reset.

In this way, a “warning” event (like an interrupt) could be detected as well as a “shutdown” event (at a higher temperature).

Care must be taken to include a timeout during the clearing of the interrupt described in the example above. The Temp Sensor has hysteresis and a timeout may be necessary to indicate if the interrupt flag has not cleared in a reasonable amount of time. If this is the case, then an error should be indicated.

Figure 1483. Temperature Sensor Configuration Register (CTL_TD)

Offset: PMC_BASE + 0x030C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS3IE	TS2IE	TS0IE	0	TRIM_ADJ_OVER[3:0]				0	0	TRIM_ADJ_UNDER[3:0]				DOUT_EN	AOUT_EN
W	TS3IE	TS2IE	TS0IE		TRIM_ADJ_OVER[3:0]						TRIM_ADJ_UNDER[3:0]				DOUT_EN	AOUT_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 1392. CTL_TD field descriptions

Field	Description
0–15	Reserved.
16 TS3IE	TS3IE Temperature Sensor input 3 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
17 TS2IE	TS2IE Temperature Sensor input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
18 TS0IE	TS0IE Temperature Sensor input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
19	Reserved.
20–23 TRIM_ADJ_OVER[3:0]	TRIM_ADJ_OVER[3:0]. Customer adjustable over trim register. These bits are a signed binary number that is added to the over temperature trim (NT_TD2) value. The TRIM value is protected from overflows and underflows due to this addition.
24–25	Reserved.
26–29 TRIM_ADJ_UNDER[3:0]	TRIM_ADJ_UNDER[3:0]. Customer adjustable under trim register. These bits are a signed binary number that is added to the Under temperature trim value (NT_TD0). The TRIM value is protected from overflows and underflows due to this addition.
30 DOUT_EN	DOUT_EN. Digital Output Enable.

Table 1392. CTL_TD field descriptions (continued)

Field	Description
31 AOUT_EN	AOUT_EN. Analog Output Enable.

60.3.1.48 Temp Sensor FCCU Event Enable Register (FEE_TD)

This Temp Sensor FCCU Event Enable Register controls whether the temperature sensor detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit (EPR_TD) is enabled, the temperature sensor detect assertion causes an FCCU event to be generated when the selected temperature exceeds the desired value.

Figure 1484. Temp Sensor FCCU Event Enable Register (FEE_TD)

Offset: PMC_BASE + 0x0318

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FEE_TS3	FEE_TS2	0	FEE_TS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Table 1393. Temp Sensor FEE_TD field descriptions

Field	Description
0–27	Reserved.
28 FEE_TS3	FEE_TS3 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
29 FEE_TS2	FEE_TS2 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
30	Reserved.
31 FEE_TS0	FEE_TS0 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.

60.3.1.49 Voltage Detect User Mode Test Register (VD_UTST)

This register is used for testing of the LVDs and HVDs during user mode.

Figure 1485. Voltage Detect User Mode Test Register (VD_UTST)

Offset: PMC_BASE + 0x0340

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	VD_UTST[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1394. VD_UTST field descriptions

Field	Description
0–25	Reserved.
26–31 VD_UTST[5:0]	VD_UTST[5:0]. User mode Test bits for testing of LVDs and HVDs.

60.3.1.50 ADC Channel Select Register (ADC_CH)

This register configures the analog channel select mux of the PMC module.

Figure 1486. ADC Channel Select Register (ADC_CS)

Offset: PMC_BASE + 0x0344

Access: Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ADC_CS					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1395. ADC_CH Bit Descriptions

Field	Description
0–25	Reserved.
26–31 ADC_CS	ADC_CS Channel Select. These bits are used to control the analog PMC's test mux going to the ADC.

60.3.1.51 Voltage Regulator 1.2V Control Register (VREG1P2_CTRL)

This register is used to disable the Auxiliary Regulator (also known as Spike or VREG1P2 Regulator) during an intentional power down event. The LSB of this register is a bit that is OR'd with the DCF control for the PMC_VREG1P2_ENB. If either the DCF loaded bit or the LSB of VREG1P2_CTRL is a 1, then the Auxiliary Regulator is disabled.

This bit should be written high before any expected power down events.

Figure 1487. Voltage Regulator 1.2V Control Register (VREG1P2_CTRL)

Offset: PMC_BASE + 0x034C												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VREG1P2_ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1396. VREG1P2_CTRL field descriptions

Field	Description
0–25	Reserved.
31 VREG1P2_ENB	VREG1P2_ENB. This bit controls if the AUX Regulator (Vreg1p2) is disabled or not. During Power Down, this bit should be written to a 1 so that the AUX regulator is disabled. By default, the DCF loaded value (PMC_VREG1P2_ENB) is used to control the AUX regulator. 0 The enable of the VREG1P2 follows the DCF loaded bit. 1 The VREG1P2 is disabled.

60.3.1.52 Module Control Register (MCR)

This register contains several status and control bits. First, it the bit that is used to detect the status and to clear the Brown Out Detection logic for the Standby RAM module. This register also contains the enables that are used to enable and disable the PMC analog block

busses. One of the enables is used for the ADC_CH_SEL bus and the other is an enable for the User Self Test bus.

The proper procedure to enable these busses would be to write the value to the register containing the bus, then write this enable high, evaluate the test associated with this bus, and then write this bus enable bit low. After that, any further tests can be continued.

Figure 1488. Module Control Register (MCR)

Offset: PMC_BASE + 0x0350

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADC_CHSEL_EN	USER_SELF_TEST_EN	0	0	0	0	0	0	TEST_PIN	0	0	0	0	0	0	SBRAM_BRDET
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1397. MCR field descriptions

Field	Description
0–15	Reserved.
16 ADC_CHSEL_EN	ADC_CHSEL_EN. This bit allows the ADC_CS bits in the ADC_CH register to be enabled. This bit should be set after the ADC_CS bus has been updated, and cleared after the test has completed. The ADC_CS bus should not be updated while the ADC_CHSEL_EN is enabled (high) 0 The ADC_CS bus is not enabled in the PMC analog. 1 The ADC_CS bus is enabled in the PMC analog.
17 USER_SELF_TEST_EN	USER_SELF_TEST_EN. This bit allows the VD_UTST bits in the VD_UTST register to be enabled. This bit should be set after the VD_UTST bus has been updated, and cleared after the test has completed. The VD_UTST bus should not be updated while the USER_SELF_TEST_EN is enabled (high) 0 The VD_UTST bus is not enabled in the PMC analog. 1 The VD_UTST bus is enabled in the PMC analog.

Table 1397. MCR field descriptions (continued)

Field	Description
18–23	Reserved.
24 TEST_PIN	TEST_PIN. This bit indicates the current value of the Test Pin. 0 Test Pin is not asserted. 1 Test Pin is asserted.
25–30	Reserved.
31 SBRAM_BRDET	SBRAM_BRDET. This bit is the detection status flag for the Standby RAM Brown Out Detection circuit. When a brown out event is detected in the Standby RAM logic, this signal asserts. In order to clear this flag, the brown out detection logic must no longer be detecting an event, and a one must be written to this bit location. 0 Currently no occurrence. 1 Standby RAM Brown Out occurrence detected.

60.4 Analog PMC interface

The outputs from the PMC digital to the PMC analog circuit consists of trim registers for each LVD circuit, an enable for each LVD (tied to the enabled state), and ADC channel select controls. The ADC channel select controls are described in more detail in the ADC interface section of this chapter, and in the Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

The signals from the PMC analog to the PMC digital logic consists of simply POR and LVD event indication signals (both active high and active low, but only active low are used). These signals are used by the PMC digital logic to create the appropriate resets for the rest of the system.

There is a settling time for the analog voltage detect circuits specified at 5 microseconds. The PMC digital counts clock edges after the loading of new flash trim values to verify that this time has passed. This time can vary from 5 microseconds to 11.4 microseconds.

60.5 Temperature Sensor Interface Logic

The Temperature Sensor interface consists of two enables: an analog and digital enable, a set of mode registers, two 6-bit trim registers (sometimes referenced as one 12-bit trim register), and an overtemp threshold select control.

The Temperature Sensor simply outputs three signals to the PMC digital logic: over_temp1, over_temp2, and under_temp signals. These signals are used by the PMC digital logic to create the appropriate resets for the rest of the system.

There are two 4-bit signed registers (TRIM_ADJ_OVER[3:0] and TRIM_ADJ_UNDER[3:0]) that can be used by software to adjust the value of the Temp Sensor Trim values. The fourth bit of these registers indicates the sign.

TRIM_ADJ_OVER is added to TS2_TRIM and TRIM_ADJ_UNDER is added to TS0_TRIM. This addition is protected from overflow and underflow (the TRIM value will simply stay at max or min value). The test interface is discussed in the test section.

For more information regarding the Temperature Sensor function, please refer to the Temperature Sensor chapter.

60.6 Standby RAM Regulator Interface Logic

The Standby RAM Regulator interface consists of one bit that indicates that a Standby RAM Brown-out event has occurred, and a method to clear this indication (by writing a one to the flag bit).

For more information regarding the Standby RAM Regulator function, please refer to the Standby RAM Regulator Analog Chapter.

60.7 Power On Reset (MC_RGM Phase Gates)

The PMC digital logic interfaces to the Reset Generation Module in order to indicate the type and conditions of the resets from the PMC and Temperature Sensor Analog blocks.

There are two signals for each the PMC and Temp Sensor blocks; one is used to indicate a destructive reset, and one is used to indicate a functional reset.

The PMC digital logic also indicates when is ok to for the RGM to allow transition to the next POR phase.

The digital PMC also uses signals from the RGM to indicate when some of the reset signals should be masked during the POR sequence.

During the POR phase 3 exit, the PMC digital logic uses all the enabled Voltage Detects to determine if it is correct to exit from Phase 3. Once all the enabled HVD/LVDs have become deasserted and the time from the LVD/HVD circuits to adjust to new trim values has elapsed, then the signal is sent to the RGM that the correct voltage levels have been achieved to exit from Phase 3.

There are nine POR signals, POR_VDDREG, POR_VDDC, POR081_C, POR085_C, POR085_F, POR098C, POR098F, POR250_F, and POR260C.

60.8 Flash interface (DCF client usage)

The PMC digital is updated during POR from the Flash block via a DCF client. This load is used for several pieces of data, but most of the data is trim values being used for analog modules (PMC, Temperature Sensor, and ADC Bandgap).

These trim bits that are loaded from the flash are not readable via the IP bus. These trim values loaded via the DCF client are not compacted (they are written individually, and not several at a time).

The 7th bit must be written in the DCF location of the VREG1P2 flash location. This location is for the VREG1p2 enable bit to be loaded from the DCF, and without programming it, the Aux regulator may be inadvertently disabled or left enabled.

A DCF client is used to interface to the flash for these early data accesses. This client also performs triple voting to keep the data safe from stray bit flips. If there are any issues occur during any of the DCF data loads (like a second write of different data), the DCF client will indicate an error to the FCCU.

Another set of data that is loaded during the DCF single read process are updated values for the REE and RES bits. Please see section [Section 60.3.1.45, Reset Event Enable Register \(REE_TD\)](#) for more information regarding the REE/RES bits load from flash.

If the REE bit is loaded via the DCF as set (enabled), then it is not possible to clear this REE bit. The reset value of the REE DCF bits are all low (thus to allow the bits to be rewriteable), but the reset value of the REE register bits are all high (to enable the resets normally out of POR).

The REE DCF bits loaded are compacted into a single DCF write (they do not match the IPS Bus register bit locations). The REE DCF loaded bits do not have the same limitation that they can only be written once and any further writes must be to the same value. The REE DCF bits can be rewritten to a new value during a new reset event.

When the REE DCF bits are updated, the REE and RES register bits are updated at the same time to the new value.

All of the DCF clients are reset via the destructive reset signal.

For more information regarding the DCF records, please refer to the DCF chapter.

The trim data is read out of the flash's differential read space (slower access, but less issues with voltage variances). Once the differential reads are completed, the flash indicates this to the PMC digital via the SSCM module.

Once the PMC trim values have been read, the analog circuit has been given time to update to the new trim values, and the LVDs have all deasserted, a signal is asserted to the flash to indicate that the voltage is high enough for faster single access reads.

All of the LVD/HVD signals must indicate that a valid voltage has been reached before the voltage indication will be sent to the flash.

For more information regarding the Flash differential reads and voltage levels required, please refer to the Flash chapter.

60.9 Other module interfaces

The FCCU block receives several signals to indicate a fault has occurred. The first is a safety error from the DCF interface indicating that some error has happened during the load of the trim data from the flash during POR. All of the DCF client safety error signals are combined in the PMC digital logic and then sent to the FCCU.

The second set of signals indicate that an HVD, LVD, or temperature event has occurred.

The LVDs that are sent to the FCCU are LVD108_CLVD114_H, LVD108_FLVD114_F, LVD108_PLVD114_P, LVD112_CLVD114_C, LVD270_CLVD280_C, LVD270_FLVD280_F, LVD270_IFLVD280_IF, LVD270_IJLVD280_IJ, LVD270_IMLVD280_IM, LVD270_OLVD280_O, LVD295_FLVD300_F, LVD360_IM, LVD400_IM, and LVD400_A.

LVD108_CLVD108_FLVD108_PLVD112_CLVD270_CLVD270_FLVD270_IFLVD270_IJLVD270_IMLVD295_A
The HVDs that are sent to the FCCU are HVD140_C, HVD145_F, HVD145_C, HVD360_F, HVD600_C, and HVD600_A.

The Temp Sensor outputs tmpsns3, tmpsns2, and tmpsns0 are all combined and sent to the FCCU.

For more information regarding the FCCU block and these signals, please refer to the FCCU chapter.

The HSM module receives four signals from the PMC digital. There are two signals that are sent to the HSM that includes both over_temp1 and over_temp2 values. The under_temp signal is also sent to the HSM as well. The last signal indicates that the PMC has detected a voltage higher than 1.4V on the internal cold detection point.

For more information regarding the HSM block and these signals, please refer to the HSM chapter of the SPC57EM80xx Security Reference Manual.

61 Power Control Unit (MC_PCU)

61.1 Introduction

The MC_PCU module plays a different role in SPC57EM80xx microcontrollers than it does in other microcontrollers. Instead of directly providing an interface to configure and control power distribution to various segments ("power domains") of the chip, the MC_PCU as an integration module.

The SPC57EM80xx microcontroller has a single power domain, Power Domain 0, which is managed by an analog Power Management Controller (PMC) and a PMC digital interface (PMC_dig). The MC_PCU address space consists of a single power domain status register and the PMC_dig registers used to configure and control the analog PMC.

Note: The PMC_dig module also includes a temperature sensor.

[Figure 1489](#) depicts the MC_PCU block diagram.

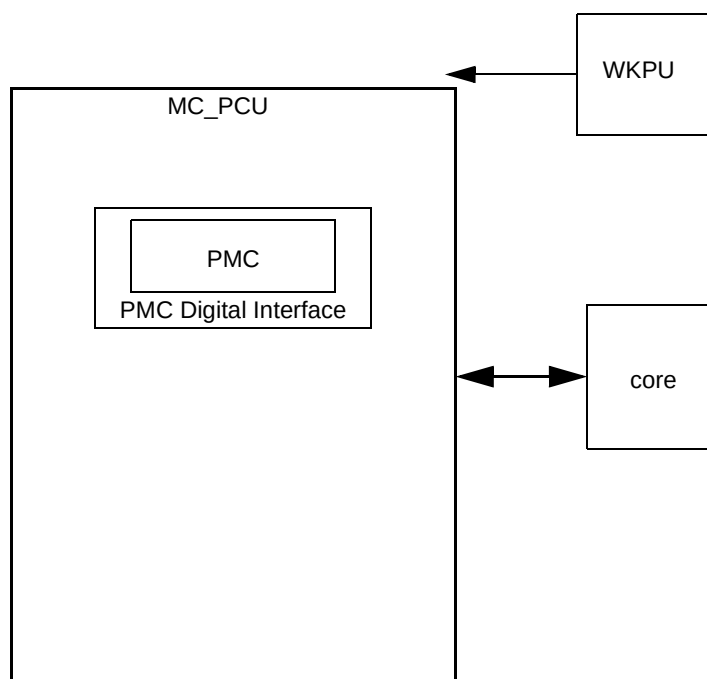


Figure 1489. MC_PCU block diagram

61.2 Memory map and register definition

61.2.1 Memory map

Table 1398. MC_PCU registers

Offset	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x0000 – 0x003C	Reserved						
0x0040	PCU_PSTAT	Power Domain Status Register	word	read	read	read	<i>on page 2655</i>
0x0044 – 0x0396	Reserved						
0x0400 – 0x076C	PMC_dig registers						
0x0800 – 0x3FFF	Reserved						

Note: Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

61.2.2 Register descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the PD0 field of the PCU_PSTAT register may be accessed as a word at offset 0x0040, as a half-word at offset 0x0042, or as a byte at offset 0x0043.

61.2.2.1 Power Domain Status Register (PCU_PSTAT)

Figure 1490. Power Domain Status Register (PCU_PSTAT)

Offset 0x0040								Access: User read, Supervisor read, Test read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

This register reflects the power status of all available power domains.

Table 1399. Power Domain Status Register (PCU_PSTAT) Field Descriptions

Field	Description
PD0	Power status for power domain #n 0Power domain is inoperable 1Power domain is operable

61.3 Functional description

Please see the PMC_dig chapter for specific details on the PMC control registers.

62 Mode Entry Module (MC_ME)

62.1 Introduction

62.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

[Figure 1491](#) shows the MC_ME block diagram.

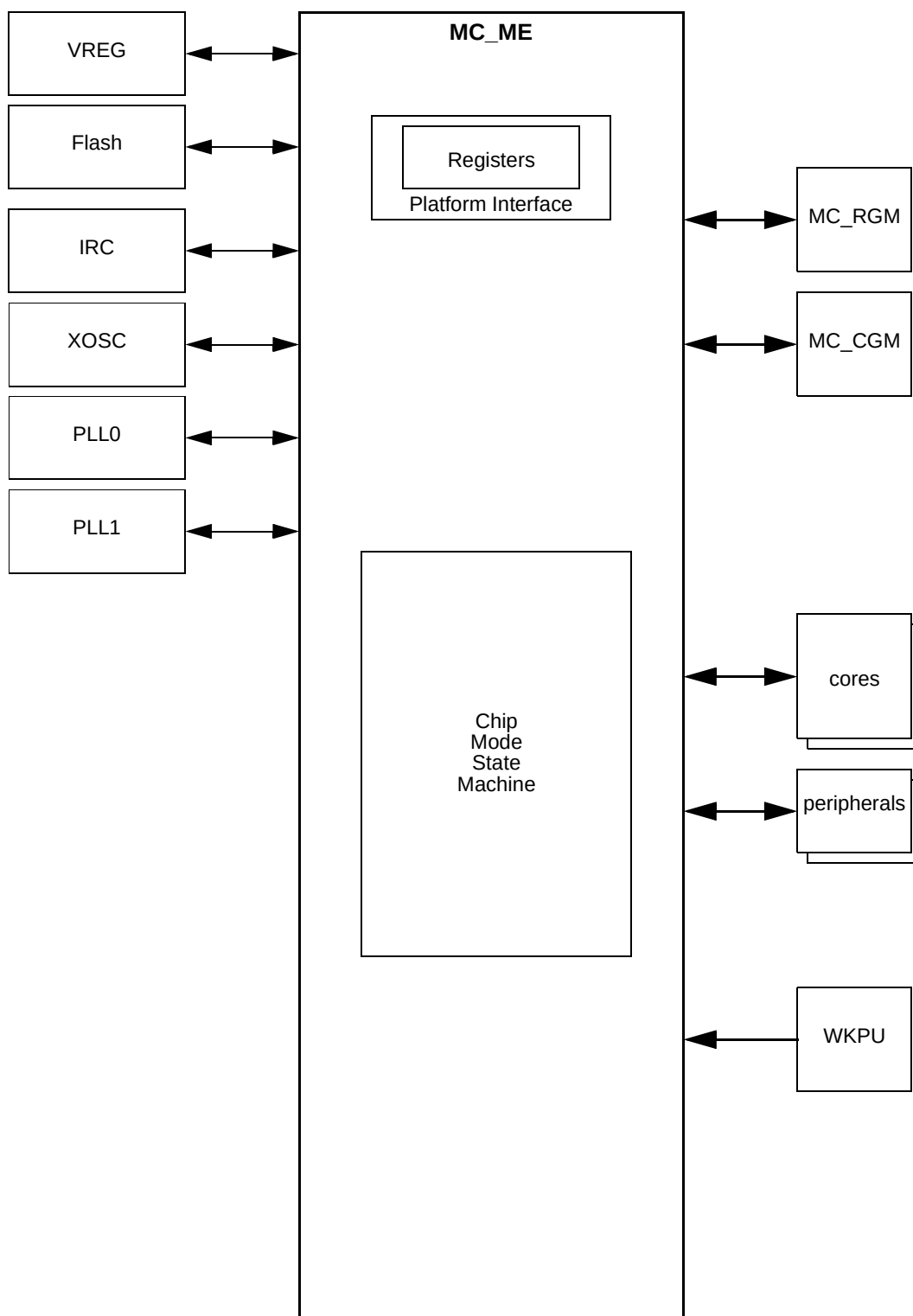


Figure 1491. MC_ME block diagram

62.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the ME_ME register
- definition of various chip mode configurations by the ME_<mode>_MC registers
- control of the actual chip mode by the ME_MCTL register
- capture of the current mode and various resource status within the contents of the ME_GS register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the ME_RUN_PC0...7, ME_LP_PC0...7, and ME_PCTLn registers
- additional core clock gating and boot address control based on the ME_CCTLn and ME_CADDRn registers
- capture of current peripheral and additional core clock gated/enabled status
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

62.1.3 Modes of operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as RESET, DRUN, SAFE, and TEST. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as RUN0...3, HALT0, and STOP0 which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, SAFE, TEST, and RUN0...3 are the chip software running modes.

[Table 1400](#) describes the MC_ME modes.

Table 1400. MC_ME mode descriptions

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAF, when present, is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from SAFE, TEST and RUN0...3	system reset assertion, RUN0...3, TEST via software, SAFE via software or hardware failure.

Table 1400. MC_ME mode descriptions (continued)

Name	Description	Entry	Exit
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN, TEST, and RUN0...3	system reset assertion, DRUN via software
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from DRUN	system reset assertion, DRUN via software
RUN0...3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from DRUN or other RUN0...3, interrupt event from HALT0, interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, other RUN0...3 modes, HALT0, STOP0 via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals like clock sources, flash, main regulator, etc. for efficient power management at the cost of higher wakeup latency.	software request from RUN0...3	system reset assertion, SAFE on hardware failure, RUN0...3 on interrupt event
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0...3	system reset assertion, SAFE on hardware failure, RUN0...3 on interrupt event or wakeup event

62.2 External signal description

The MC_ME has no connections to any external pins.

62.3 Memory map and register definition

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of additional core enabling and disabling control and status reporting

62.3.1 Memory map

Table 1401. MC_ME register description

Address offset	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x000	ME_GS	Global Status	word	read	read	read	on page 2672
0x004	ME_MCTL	Mode Control	word	read	read/write	read/write	on page 2674
0x008	ME_ME	Mode Enable	word	read	read/write	read/write	on page 2675
0x00C	ME_IS	Interrupt Status	word	read	read/write	read/write	on page 2677
0x010	ME_IM	Interrupt Mask	word	read	read/write	read/write	on page 2678
0x014	ME_IMTS	Invalid Mode Transition Status	word	read	read/write	read/write	on page 2679
0x018	ME_DMTS	Debug Mode Transition Status	word	read	read	read	on page 2680
0x020	ME_RESET_MC	RESET Mode Configuration	word	read	read	read	on page 2684
0x024	ME_TEST_MC	TEST Mode Configuration	word	read	read/write	read/write	on page 2685
0x028	ME_SAFE_MC	SAFE Mode Configuration	word	read	read/write	read/write	on page 2685
0x02C	ME_DRUN_MC	DRUN Mode Configuration	word	read	read/write	read/write	on page 2685
0x030	ME_RUN0_MC	RUN0 Mode Configuration	word	read	read/write	read/write	on page 2686
0x034	ME_RUN1_MC	RUN1 Mode Configuration	word	read	read/write	read/write	on page 2686
0x038	ME_RUN2_MC	RUN2 Mode Configuration	word	read	read/write	read/write	on page 2686
0x03C	ME_RUN3_MC	RUN3 Mode Configuration	word	read	read/write	read/write	on page 2686
0x040	ME_HALTO_MC	HALTO Mode Configuration	word	read	read/write	read/write	on page 2686
0x048	ME_STOP0_MC	STOP0 Mode Configuration	word	read	read/write	read/write	on page 2687
0x060	ME_PS0	Peripheral Status 0	word	read	read	read	on page 2689
0x064	ME_PS1	Peripheral Status 1	word	read	read	read	on page 2690

Table 1401. MC_ME register description (continued)

Address offset	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x068	ME_PS2	Peripheral Status 2	word	read	read	read	on page 2691
0x06C	ME_PS3	Peripheral Status 3	word	read	read	read	on page 2692
0x070	ME_PS4	Peripheral Status 4	word	read	read	read	on page 2692
0x074	ME_PS5	Peripheral Status 5	word	read	read	read	on page 2693
0x078	ME_PS6	Peripheral Status 6	word	read	read	read	on page 2693
0x07C	ME_PS7	Peripheral Status 7	word	read	read	read	on page 2694
0x080	ME_RUN_PC0	Run Peripheral Configuration 0	word	read	read/write	read/write	on page 2695
0x084	ME_RUN_PC1	Run Peripheral Configuration 1	word	read	read/write	read/write	on page 2695
...							
0x09C	ME_RUN_PC7	Run Peripheral Configuration 7	word	read	read/write	read/write	on page 2695
0x0A0	ME_LP_PC0	Low-Power Peripheral Configuration 0	word	read	read/write	read/write	on page 2696
0x0A4	ME_LP_PC1	Low-Power Peripheral Configuration 1	word	read	read/write	read/write	on page 2696
...							
0x0BC	ME_LP_PC7	Low-Power Peripheral Configuration 7	word	read	read/write	read/write	on page 2696
0x0C9	ME_PCTL9	LFAST_0 Control	byte	read	read/write	read/write	on page 2696
0x0DE	ME_PCTL30	PIT_RTC_0 Control	byte	read	read/write	read/write	on page 2696
0x0DF	ME_PCTL31	PIT_RTC_1 Control	byte	read	read/write	read/write	on page 2696
0x0E4	ME_PCTL36	DMAMUX_0 Control	byte	read	read/write	read/write	on page 2696
0x0E6	ME_PCTL38	CRC_0 Control	byte	read	read/write	read/write	on page 2696
0x0FA	ME_PCTL58	ADCSD_4 Control	byte	read	read/write	read/write	on page 2696
0x0FB	ME_PCTL59	ADCSD_2 Control	byte	read	read/write	read/write	on page 2696

Table 1401. MC_ME register description (continued)

Address offset	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x0FC	ME_PCTL60	ADCSD_0 Control	byte	read	read/write	read/write	on page 2696
0x104	ME_PCTL68	MCAN_3 Control	byte	read	read/write	read/write	on page 2696
0x105	ME_PCTL69	MCAN_2 Control	byte	read	read/write	read/write	on page 2696
0x106	ME_PCTL70	MCAN_1 Control	byte	read	read/write	read/write	on page 2696
0x108	ME_PCTL72	TTCAN Control	byte	read	read/write	read/write	on page 2696
0x10A	ME_PCTL74	CAN_RAM_CTRL Control	byte	read	read/write	read/write	on page 2696
0x115	ME_PCTL85	LINFlexD_14 Control	byte	read	read/write	read/write	on page 2696
0x11B	ME_PCTL91	LINFlexD_1 Control	byte	read	read/write	read/write	on page 2696
0x11C	ME_PCTL92	LINFlexD_0 Control	byte	read	read/write	read/write	on page 2696
0x11D	ME_PCTL93	DSPI_12 Control	byte	read	read/write	read/write	on page 2696
0x121	ME_PCTL97	DSPI_4 Control	byte	read	read/write	read/write	on page 2696
0x122	ME_PCTL98	DSPI_1 Control	byte	read	read/write	read/write	on page 2696
0x123	ME_PCTL99	DSPI_0 Control	byte	read	read/write	read/write	on page 2696
0x125	ME_PCTL101	IIC_0 Control	byte	read	read/write	read/write	on page 2696
0x128	ME_PCTL104	SENT_0 Control	byte	read	read/write	read/write	on page 2696
0x12B	ME_PCTL107	FLEXRAY Control	byte	read	read/write	read/write	on page 2696
0x12F	ME_PCTL111	PSI5_0 Control	byte	read	read/write	read/write	on page 2696
0x130	ME_PCTL112	ADCSAR_b Control	byte	read	read/write	read/write	on page 2696
0x13B	ME_PCTL123	ADCSAR_4 Control	byte	read	read/write	read/write	on page 2696
0x13F	ME_PCTL127	ADCSAR_0 Control	byte	read	read/write	read/write	on page 2696

Table 1401. MC_ME register description (continued)

Address offset	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x140	ME_PCTL128	GTMIN_T Control	byte	read	read/write	read/write	on page 2696
0x166	ME_PCTL166	CRC_1 Control	byte	read	read/write	read/write	on page 2696
0x17A	ME_PCTL186	ADCSD_5 Control	byte	read	read/write	read/write	on page 2696
0x17B	ME_PCTL187	ADCSD_3 Control	byte	read	read/write	read/write	on page 2696
0x17C	ME_PCTL188	ADCSD_1 Control	byte	read	read/write	read/write	on page 2696
0x195	ME_PCTL213	LINFlexD_15 Control	byte	read	read/write	read/write	on page 2696
0x19C	ME_PCTL220	LINFlexD_2 Control	byte	read	read/write	read/write	on page 2696
0x1A1	ME_PCTL225	DSPI_5 Control	byte	read	read/write	read/write	on page 2696
0x1A2	ME_PCTL226	DSPI_3 Control	byte	read	read/write	read/write	on page 2696
0x1A3	ME_PCTL227	DSPI_2 Control	byte	read	read/write	read/write	on page 2696
0x1A8	ME_PCTL232	SENT_1 Control	byte	read	read/write	read/write	on page 2696
0x1AF	ME_PCTL239	PSI5_1 Control	byte	read	read/write	read/write	on page 2696
0x1B8	ME_PCTL248	ADCSAR_7 Control	byte	read	read/write	read/write	on page 2696
0x1B9	ME_PCTL249	ADCSAR_6 Control	byte	read	read/write	read/write	on page 2696
0x1BC	ME_PCTL252	ADCSAR_3 Control	byte	read	read/write	read/write	on page 2696
0x1BD	ME_PCTL253	ADCSAR_2 Control	byte	read	read/write	read/write	on page 2696
0x1BE	ME_PCTL254	ADCSAR_1 Control	byte	read	read/write	read/write	on page 2696
0x1C0	ME_CS	Core Status	word	read	read	read	on page 2698
0x1C4	ME_CCTL0	CORE0 Control	halfword	read/write	read/write	read/write	on page 2698
0x1C6	ME_CCTL1	CORE1 Control	halfword	read/write	read/write	read/write	on page 2699

Table 1401. MC_ME register description (continued)

Address offset	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x1C8	ME_CCTL2	CORE2 Control	halfword	read/write	read/write	read/write	on page 2701
0x1CA	ME_CCTL3	CORE3 Control	halfword	read/write	read/write	read/write	on page 2701
0x1CC	ME_CCTL4	CORE4 Control	halfword	read/write	read/write	read/write	on page 2702
0x1E0	ME_CADDR0	CORE0 Address	word	read/write	read/write	read/write	on page 2703
0x1E4	ME_CADDR1	CORE1 Address	word	read/write	read/write	read/write	on page 2704
0x1EC	ME_CADDR3	CORE3 Address	word	read/write	read/write	read/write	on page 2705
0x1F0	ME_CADDR4	CORE4 Address	word	read/write	read/write	read/write	on page 2706

Note: Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Table 1402. MC_ME memory map

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x000	ME_GS	R	S_CURRENT_MODE				S_MTRANS	1	0	0	S_PDO	0	0	S_MVR	0	0	S_FLAS	
		W																
		R	0	0	0	0	0	0	0	0	S_PLL1	S_PLLO	S_XOSC	S_IRC	S_SYSCCLK			
		W																
0x004	ME_MCTL	R	TARGET_MODE				0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
		W	KEY															

Table 1402. MC_ME memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x008	ME_ME	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	RESET_DEST	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET_FUNC
		W																
0x00C	ME_IS	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	I_ICONF_CC	I_ICONF_CU	I_ICONF	I_IMODE	I_SAFE	I_MTC
		W											w1c	w1c	w1c	w1c	w1c	w1c
0x010	ME_IM	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	M_ICONF_CC	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC
		W																
0x014	ME_IMTS	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	S_MRIG	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
		W											w1c	w1c	w1c	w1c	w1c	w1c

Table 1402. MC_ME memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x018	ME_DMTS	R	PREVIOUS_MODE				0	0	0	0	MPH_BUSY	0	0	PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
		W																
		R	CDP_PRPH_0_255	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	0	FLASH_SC	CDP_PRPH_224_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
		W																
0x01C	reserved																	
0x020	ME_RESET_MC	R	PWRLVL			0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
		W																
		R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
		W																
0x024	ME_TEST_MC	R	PWRLVL			0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
		W																
		R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
		W																
0x028	ME_SAFE_MC	R	PWRLVL			0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
		W																
		R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
		W																

Table 1402. MC_ME memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x02C	ME_DRUN_MC	R	PWRLVL			0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
		W																
		R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
		W																
0x030 ... 0x03C	ME_RUN0...3_MC	R	PWRLVL			0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
		W																
		R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
		W																
0x040	ME_HALTO_MC	R	PWRLVL			0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
		W																
		R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
		W																
0x044	reserved																	
0x048	ME_STOP0_MC	R	PWRLVL			0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
		W																
		R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
		W																
0x04C ... 0x05C	reserved																	

Table 1402. MC_ME memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x060	ME_PS0	R	S_PIT_RTC_1	S_PIT_RTC_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	S_SPLI_0	0	S_LFAST_0	0	0	0	0	0	0	0	0	0
		W																
0x064	ME_PS1	R	0	0	0	S_ADCSD_0	S_ADCSD_2	S_ADCSD_4	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	S_CRC_0	0	S_DMAMUX_0	0	0	0	0
		W																
0x068	ME_PS2	R	0	0	S_DSPI_12	S_LINFlexD_0	S_LINFlexD_1	0	0	0	0	0	S_LINFlexD_14	0	0	0	0	0
		W																
		R	0	0	0	0	0	S_CAN_RAM_CTRL	0	S_TTCAN	0	S_MCAN_1	S_MCAN_2	S_MCAN_3	0	0	0	0
		W																

Table 1402. MC_ME memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x06C	ME_PS3	R	S_ADCSAR_0	0	0	0	S_ADCSAR_4	0	0	0	0	0	0	0	0	0	0	S_ADCSAR_b
		W																
		R	S_PS15_0	0	0	0	S_FLEXRAY	0	0	S_SENT_0	0	0	S_IIC_0	0	S_DSPI_0	S_DSPI_1	S_DSPI_4	0
		W																
0x070	ME_PS4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	S_GTMINT
		W																
0x074	ME_PS5	R	0	0	0	S_ADCSD_1	S_ADCSD_3	S_ADCSD_5	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	S_CRC_1	0	0	0	0	0	0
		W																
0x078	ME_PS6	R	0	0	0	S_LINFlexD_2	0	0	0	0	0	0	S_LINFlexD_15	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																

Table 1402. MC_ME memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x07C	ME_PS7	R	0	S_ADCSAR_1	S_ADCSAR_2	S_ADCSAR_3	0	0	S_ADCSAR_6	S_ADCSAR_7	0	0	0	0	0	0	0	0
		W																
		R	S_PS15_1	0	0	0	0	0	0	S_SENT_1	0	0	0	0	S_DSPI_2	S_DSPI_3	S_DSPI_5	0
		W																
0x080 ... 0x09C	ME_RUN_PC0...7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
		W																
0x0A0 ... 0x0BC	ME_LP_PC0...7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	STOP0	0	HALT0	0	0	0	0	0	0	0	0
		W																
0x0C0 ... 0x1BC	ME_PCTL0...255 (1)	R	0	DBG_F	LP_CFG			RUN_CFG			0	DBG_F	LP_CFG			RUN_CFG		
		W																
		R	0	DBG_F	LP_CFG			RUN_CFG			0	DBG_F	LP_CFG			RUN_CFG		
		W																
0x1C0	ME_CS	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	S_CORE4	S_CORE3	S_CORE2	S_CORE1	S_CORE0
		W																

Table 1402. MC_ME memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0x1C4	ME_CCTL0...1	R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET	
		W																	
		R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET	
		W																	
0x1C8	ME_CCTL2...3	R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET	
		W																	
		R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET	
		W																	
0x1CC	ME_CCTL4	R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET	
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
0x1D0 ... 0x1DC	reserved																		
0x1E0	ME_CADDR0	R	ADDR[31:16]																RMC
		W																	
		R	ADDR[15:2]														0		
		W																	
0x1E4	ME_CADDR1	R	ADDR[31:16]																RMC
		W																	
		R	ADDR[15:2]														0		
		W																	

Table 1402. MC_ME memory map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0x1EC	ME_CADDR3	R	ADDR[31:16]																
		W																	
		R	ADDR[15:2]														0	RMC	
		W																	
0x1F0	ME_CADDR4	R	ADDR[31:16]																
		W																	
		R	ADDR[15:2]														0	RMC	
		W																	

1. There is space in the register map for 256 peripherals. Please see [Table 1401](#) for the ME_PCTLn locations actually occupied. The unoccupied locations contain a read-only byte value of 0x00.

62.3.2 Register description

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

62.3.2.1 Global Status Register (ME_GS)

This register contains global mode status.

Figure 1492. Global Status Register (ME_GS)

Address 0x000				Access: User read, Supervisor read, Test read															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	S_CURRENT_MODE				S_MTRANS	1	0	0	S_PDO	0	0	S_MVR	0	0	S_FLA				
W																			
Reset	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	S_PLL1	S_PLL0	S_XOSC	S_IRC	S_SYSCCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Table 1403. Global Status Register (ME_GS) Field Descriptions

Field	Description
S_CURRENT_MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
S_PDO	Output power-down status — This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In STOP0 mode, the slew rate control mechanism of the output pads is disabled to reduce power consumption, but the state of the output remains functional.
S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
S_FLA	Flash availability status 00 Flash is not available 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode and available for use
S_PLL1	secondary PLL status 0 secondary PLL is not stable 1 secondary PLL is providing a stable clock
S_PLL0	primary PLL status 0 primary PLL is not stable 1 primary PLL is providing a stable clock
S_XOSC	external crystal oscillator status 0 external crystal oscillator is not stable 1 external crystal oscillator is providing a stable clock

Table 1403. Global Status Register (ME_GS) Field Descriptions (continued)

Field	Description
S_IRC	16 MHz internal RC oscillator status 0 16 MHz internal RC oscillator is not stable 1 16 MHz internal RC oscillator is providing a stable clock
S_SYSClk	System clock switch status — These bits specify the system clock currently used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

62.3.2.2 Mode Control Register (ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<mode>_MC registers must respect this for successful mode requests.

Note: *Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.*

Figure 1493. Mode Control Register (ME_MCTL)

Address 0x004

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TARGET_MODE				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
W	KEY															
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Table 1404. Mode Control Register (ME_MCTL) Field Descriptions

Field	Description
TARGET MODE	Target chip mode — These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0...3 mode value. 0000 RESET (triggers a 'functional' reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 RESET (triggers a 'destructive' reset event)
KEY	Control key — These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101101011110000 (0x5AF0) INVERTED KEY: 1010010100001111 (0xA50F)

62.3.2.3 Mode Enable Register (ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. RESET, SAFE, DRUN, and RUN0 modes are always enabled.

Figure 1494. Mode Enable Register (ME_ME)

Address 0x008

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESET_DEST	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET_FUNC
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1

Table 1405. Mode Enable Register (ME_ME) Field Descriptions

Field	Description
RESET_DEST	'destructive' RESET mode enable 1 'destructive' RESET mode is enabled
STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
HALT0	HALT0 mode enable 0 HALT0 mode is disabled 1 HALT0 mode is enabled
RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
RUN0	RUN0 mode enable 1 RUN0 mode is enabled
DRUN	DRUN mode enable 1 DRUN mode is enabled
SAFE	SAFE mode enable 1 SAFE mode is enabled

Table 1405. Mode Enable Register (ME_ME) Field Descriptions (continued)

Field	Description
TEST	TEST mode enable 0 TEST mode is disabled 1 TEST mode is enabled
RESET_FUNC	'functional' RESET mode enable 1 'functional' RESET mode is enabled

62.3.2.4 Interrupt Status Register (ME_IS)

This register provides the current interrupt status.

Figure 1495. Interrupt Status Register (ME_IS)

Address 0x00C

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	I_ICONF_CC	I_ICONF_CU	I_ICONF	I_MODE	I_SAFE	I_MTC
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1406. Interrupt Status Register (ME_IS) Field Descriptions

Field	Description
I_ICONF_CC	Invalid mode configuration interrupt (core configuration) — This bit is set if a write access to one of the ME_CCTLn registers is attempted while a mode transition is in progress. 0 No write to an ME_CADDRn register was attempted during an ongoing mode transition 1 A write to an ME_CADDRn register was attempted during an ongoing mode transition
I_ICONF_CU	Invalid mode configuration interrupt (Clock Usage) — This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration (clock usage) interrupt occurred 1 Invalid mode configuration (clock usage) interrupt is pending
I_ICONF	Invalid mode configuration interrupt — This bit is set whenever a write operation to ME_<mode>_MC registers with invalid mode configuration is attempted. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending

Table 1406. Interrupt Status Register (ME_IS) Field Descriptions (continued)

Field	Description
I_IMODE	Invalid mode interrupt — This bit is set whenever an invalid mode transition is requested. It is cleared by writing a '1' to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
I_SAFE	SAFE mode interrupt — This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
I_MTC	Mode transition complete interrupt — This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, or STOP0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

62.3.2.5 Interrupt Mask Register (ME_IM)

This register controls whether an event generates an interrupt or not.

Figure 1496. Interrupt Mask Register (ME_IM)

Address 0x010

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	M_ICONF_CC	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1407. Interrupt Mask Register (ME_IM) Field Descriptions

Field	Description
M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled

Table 1407. Interrupt Mask Register (ME_IM) Field Descriptions (continued)

Field	Description
M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

62.3.2.6 Invalid Mode Transition Status Register (ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Figure 1497. Invalid Mode Transition Status Register (ME_IMTS)

Address 0x014

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	S_MRIG	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1408. Invalid Mode Transition Status Register (ME_IMTS) Field Descriptions

Field	Description
S_MRIG	Mode Request Ignored status — This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress, a transition to STOP0 mode is requested while a wakeup or interrupt request is active, or a transition to HALT0 mode is requested while an interrupt request is active. It is cleared by writing a '1' to this bit. 0 Mode transition requested is not ignored 1 Mode transition requested is ignored
S_MTI	Mode Transition Illegal status — This bit is set whenever a new mode is requested while some other mode transition process is active (S_MTRANS is '1'). Please see Section 62.4.5, Mode Transition Interrupts , for the exceptions to this behavior. It is cleared by writing a '1' to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
S_MRI	Mode Request Illegal status — This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a '1' to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
S_DMA	Disabled Mode Access status — This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
S_NMA	Non-existing Mode Access status — This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
S_SEA	SAFE Event Active status — This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a '1' to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

62.3.2.7 Debug Mode Transition Status Register (ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS.S_MTRANS may be taking longer than expected.

Note: The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Figure 1498. Debug Mode Transition Status Register (ME_DMTS)

Address 0x018

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0	0	0	0	MPH_BUSY	0	0	PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDP_PRPH_0_255	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSClk_SW	reserved	FLASH_SC	CDP_PRPH_224_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1409. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions

Field	Description
PREVIOUS_MODE	Previous chip mode — These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator — This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its power-up/down sequencing. It is cleared when the MC_PCU has responded. 0 Handshake is not busy 1 Handshake is busy

Table 1409. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions (continued)

Field	Description
PMC_PROG	MC_PCU Mode Change in Progress indicator — This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
DBG_MODE	Debug mode indicator — This bit is set while the chip is in debug mode. 0 The chip is not in debug mode 1 The chip is in debug mode
CCKL_PROG	Core Clock Enable/Disable in Progress — This bit is set while any core's clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled 1 A core clock is being enabled or disabled
PCS_PROG	Progressive System Clock Switching in Progress — This bit is set while the progressive system clock switching process is in progress. 0 PCS is not in progress 1 PCS is in progress
SMR	SAFE mode request from MC_RGM is active indicator — This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared. 0 A SAFE mode request is not active 1 A SAFE mode request is active
CDP_PRPH_0_255	Clock Disable Process Pending status for Peripherals 0...255 ⁽¹⁾ — This bit is set when any peripheral appearing has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
IRC_SC	IRC State Change during mode transition indicator — This bit is set when the 16 MHz internal RC oscillator is requested to change its power up/down state. It is cleared when the 16 MHz internal RC oscillator has completed its state change. 0 No state change is taking place 1 A state change is taking place

Table 1409. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions (continued)

Field	Description
SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator — This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A secondary clock source is a clock source other than IRC.) 0 No state change is taking place 1 A state change is taking place
SYSCLK_SW	System Clock Switching pending status — 0 No system clock source switching is pending 1 A system clock source switching is pending
FLASH_SC	FLASH State Change during mode transition indicator — This bit is set when the FLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
CDP_PRPH_224_255	Clock Disable Process Pending status for Peripherals 224...255 ⁽¹⁾ — This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_192_223	Clock Disable Process Pending status for Peripherals 192...223 ² — This bit is set when any peripheral appearing in ME_PS6 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_160_191	Clock Disable Process Pending status for Peripherals 160...191 ² — This bit is set when any peripheral appearing in ME_PS5 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_128_159	Clock Disable Process Pending status for Peripherals 128...159 ² — This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_96_127	Clock Disable Process Pending status for Peripherals 96...127 ² — This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table 1409. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions (continued)

Field	Description
CDP_PRPH_64_95	Clock Disable Process Pending status for Peripherals 64...95 ² — This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_32_63	Clock Disable Process Pending status for Peripherals 32...63 ² — This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH_0_31	Clock Disable Process Pending status for Peripherals 0...31 ² — This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

1. Peripheral n corresponds to the **ME_PCTLn** register. Please refer to [Table 1401](#) for the **ME_PCTLn** locations actually occupied, which in turn indicates which peripherals are reported in the **ME_DMTS** register.

62.3.2.8 RESET Mode Configuration Register (ME_RESET_MC)

This register configures system behavior during RESET mode. See [Table 1410](#) for details.

Note: The following configuration values are set according to the chip configuration:

- XOSCON

Figure 1499. RESET Mode Configuration Register (ME_RESET_MC)

Address 0x020

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

62.3.2.9 TEST Mode Configuration Register (ME_TEST_MC)

This register configures system behavior during TEST mode. See [Table 1410](#) for details.

Note: Byte write accesses are not allowed to this register.

Figure 1500. TEST Mode Configuration Register (ME_TEST_MC)

Address 0x024				Access: User read, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON				
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

62.3.2.10 SAFE Mode Configuration Register (ME_SAFE_MC)

This register configures system behavior during SAFE mode. See [Table 1410](#) for details.

Note: Byte write accesses are not allowed to this register.

Figure 1501. SAFE Mode Configuration Register (ME_SAFE_MC)

Address 0x028				Access: User read, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON				
W																			
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1			

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

62.3.2.11 DRUN Mode Configuration Register (ME_DRUN_MC)

This register configures system behavior during DRUN mode. See [Table 1410](#) for details.

Note: Byte write accesses are not allowed to this register.

The following configuration values are set according to the chip configuration:

- XOSCON

Figure 1502. DRUN Mode Configuration Register (ME_DRUN_MC)

Address 0x02C

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

62.3.2.12 RUN0...3 Mode Configuration Registers (ME_RUN0...3_MC)

This register configures system behavior during RUN0...3 modes. See [Table 1410](#) for details.

Note: Byte write accesses are not allowed to this register.

Figure 1503. RUN0...3 Mode Configuration Registers (ME_RUN0...3_MC)

Address 0x030 - 0x03C

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

62.3.2.13 HALT0 Mode Configuration Register (ME_HALT0_MC)

This register configures system behavior during HALT0 mode. See [Table 1410](#) for details.

Note: Byte write accesses are not allowed to this register.

Figure 1504. HALT0 Mode Configuration Register (ME_HALT0_MC)

Address 0x040

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

62.3.2.14 STOP0 Mode Configuration Register (ME_STOP0_MC)

This register configures system behavior during STOP0 mode. See [Table 1410](#) for details.

Note: Byte write accesses are not allowed to this register.

Figure 1505. STOP0 Mode Configuration Register (ME_STOP0_MC)

Address 0x048

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Table 1410. Mode Configuration Registers (ME_<mode>_MC) Field Descriptions

Field	Description
PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
XOSCON	external crystal oscillator control ⁽¹⁾ 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

- For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

62.3.2.15 Peripheral Status Register 0 (ME_PS0)

This register provides the status of the peripherals. See [Table 1411](#) for details.

Figure 1506. Peripheral Status Register 0 (ME_PS0)

Address 0x060				Access: User read, Supervisor read, Test read												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_PIT_RTC_1	S_PIT_RTC_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	S_SPLI_0	0	S_LFAST_0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

62.3.2.16 Peripheral Status Register 1 (ME_PS1)

This register provides the status of the peripherals. See [Table 1411](#) for details.

Figure 1507. Peripheral Status Register 1 (ME_PS1)

Address 0x064 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_ADCSD_0	S_ADCSD_2	S_ADCSD_4	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	S_CRC_0	0	S_DMAMUX_0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

62.3.2.17 Peripheral Status Register 2 (ME_PS2)

This register provides the status of the peripherals. See [Table 1411](#) for details.

Figure 1508. Peripheral Status Register 2 (ME_PS2)

Address 0x068

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	S_DSPI_12	S_LINFlexD_0	S_LINFlexD_1	0	0	0	0	0	S_LINFlexD_14	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_CAN_RAM_CTRL	0	S_TTCAN	0	S_MCAN_1	S_MCAN_2	S_MCAN_3	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

62.3.2.18 Peripheral Status Register 3 (ME_PS3)

This register provides the status of the peripherals. See [Table 1411](#) for details.

Figure 1509. Peripheral Status Register 3 (ME_PS3)

Address 0x06C

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_ADCSAR_0	0	0	0	S_ADCSAR_4	0	0	0	0	0	0	0	0	0	0	S_ADCSAR_b
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_PSIS_0	0	0	0	S_FLEXRAY	0	0	S_SENT_0	0	0	S_IIC_0	0	S_DSPI_0	S_DSPI_1	S_DSPI_4	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

62.3.2.19 Peripheral Status Register 4 (ME_PS4)

This register provides the status of the peripherals. See [Table 1411](#) for details.

Figure 1510. Peripheral Status Register 4 (ME_PS4)

Address 0x070

Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	S_GTMINT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

62.3.2.20 Peripheral Status Register 5 (ME_PS5)

This register provides the status of the peripherals. See [Table 1411](#) for details.

Figure 1511. Peripheral Status Register 5 (ME_PS5)

Address 0x068				Access: User read, Supervisor read, Test read												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_ADCSD_1	S_ADCSD_3	S_ADCSD_5	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	S_CRC_1	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

62.3.2.21 Peripheral Status Register 6 (ME_PS6)

This register provides the status of the peripherals. See [Table 1411](#) for details.

Figure 1512. Peripheral Status Register 6 (ME_PS6)

Address 0x06C				Access: User read, Supervisor read, Test read												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_LINFlexD_2	0	0	0	0	0	0	S_LINFlexD_15	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

62.3.2.22 Peripheral Status Register 7 (ME_PS7)

This register provides the status of the peripherals.

Figure 1513. Peripheral Status Register 7 (ME_PS7)

Address 0x070								Access: User read, Supervisor read, Test read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_ADCSAR_1	S_ADCSAR_2	S_ADCSAR_3	0	0	S_ADCSAR_6	S_ADCSAR_7	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_PSI5_1	0	0	0	0	0	0	S_SENT_1	0	0	0	0	S_DSPI_2	S_DSPI_3	S_DSPI_5	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1411. Peripheral Status Registers (ME_PS_n) field descriptions

Field	Description
S_<periph>	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

62.3.2.23 Run Peripheral Configuration Registers (ME_RUN_PC0...7)

These registers configure eight different types of peripheral behavior during run modes.

Figure 1514. Run Peripheral Configuration Registers (ME_RUN_PC0...7)

Address 0x080 - 0x09C

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1412. Run Peripheral Configuration Registers (ME_RUN_PC0...7) Field Descriptions

Field	Description
RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN1	Peripheral control during RUN1 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated 1 Peripheral is active
RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

62.3.2.24 Low-Power Peripheral Configuration Registers (ME_LP_PC0...7)

These registers configure eight different types of peripheral behavior during non-run modes.

Figure 1515. Low-Power Peripheral Configuration Registers (ME_LP_PC0...7)

Address 0x0A0 - 0x0BC								Access: User read, Supervisor read/write, Test read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	STOP0	0	HALT0	0	0	0	0	0	0	0	0
W						STOP0		HALT0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1413. Low-Power Peripheral Configuration Registers (ME_LP_PC0...7) Field Descriptions

Field	Description
STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
HALT0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated 1 Peripheral is active

62.3.2.25 Peripheral Control Registers (ME_PCTLn)

These registers select the configurations during run and non-run modes for each peripheral. Please see [Table 1401](#) for information on which ME_PCTLn locations are actually occupied. The unoccupied locations contain a read-only byte value of 0x00.

Note: After modifying any of the ME_RUN_PC0...7, ME_LP_PC0...7, and ME_PCTLn registers, software must request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behavior between the peripheral clock control process and the clock status reporting in the ME_PSn registers.

Figure 1516. Peripheral Control Registers (ME_PCTLn)

Address 0x0C0 - 0x1BF								Access: User read, Supervisor read/write, Test read/write							
	0	1	2	3	4	5	6	7							
R	0														
W		DBG_F		LP_CFG				RUN_CFG							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1414. Peripheral Control Registers (ME_PCTLn) Field Descriptions

Field	Description
DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes. <i>Note: This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses.</i>
LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0...7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0...7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

62.3.2.26 Core Status Register (ME_CS)

This register provides the status of the additional cores.

Figure 1517. Core Status Register (ME_CS)

Address 0x1C0								Access: User read, Supervisor read, Test read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	S_CORE4	S_CORE3	S_CORE2	S_CORE1	S_CORE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1415. Core Status Register (ME_CS) field descriptions

Field	Description
S_CORE4	Core 4 status — This bits specifies the current status of the HSM which is controlled by the ME_CCTL4 register. 0 the HSM is disabled 1 the HSM is running
S_CORE3	Core 3 status — This bits specifies the current status of main core_1 which is controlled by the ME_CCTL3 register. 0 main core_1 is disabled 1 main core_1 is running
S_CORE2	Core 2 status — This bits specifies the current status of checker core_0s which is controlled by the ME_CCTL2 register. 0 checker core_0s is disabled 1 checker core_0s is running
S_CORE1	Core 1 status — This bits specifies the current status of main core_0 which is controlled by the ME_CCTL1 register. 0 main core_0 is disabled 1 main core_0 is running
S_CORE0	Core 0 status — This bits specifies the current status of peripheral core_2 which is controlled by the ME_CCTL0 register. 0 peripheral core_2 is disabled 1 peripheral core_2 is running

62.3.2.27 Core Control Register 0 (ME_CCTL0)

This register controls whether peripheral core_2 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] = 1).

Figure 1518. Core Control Register 0 (ME_CCTL0)

Address 0x1C4

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0

Table 1416. ME_CCTL0 field descriptions

Field	Description
STOP0	Core control during STOP0 — peripheral core_2 is always disabled during STOP0.
HALT0	Core control during HALT0 — peripheral core_2 is always disabled during HALT0.
RUN3	Core control during RUN3 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running
RUN2	Core control during RUN2 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running
RUN1	Core control during RUN1 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running
RUN0	Core control during RUN0 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
DRUN	Core control during DRUN 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
SAFE	Core control during SAFE 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
TEST	Core control during TEST 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
RESET	Core control during RESET — peripheral core_2 is always disabled during RESET.

62.3.2.28 Core Control Register (ME_CCTL1)

This register controls whether main core_0 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] = 1).

Figure 1519. Core Control Register (ME_CCTL1)

Address 0x1C6

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1417. ME_CCTL1 field descriptions

Field	Description
STOP0	Core control during STOP0 — main core_0 is always disabled during STOP0.
HALT0	Core control during HALT0 — main core_0 is always disabled during HALT0.
RUN3	Core control during RUN3 0 main core_0 is disabled with clock gated 1 main core_0 is running
RUN2	Core control during RUN2 0 main core_0 is disabled with clock gated 1 main core_0 is running
RUN1	Core control during RUN1 0 main core_0 is disabled with clock gated 1 main core_0 is running
RUN0	Core control during RUN0 0 main core_0 is frozen with clock gated 1 main core_0 is running
DRUN	Core control during DRUN 0 main core_0 is frozen with clock gated 1 main core_0 is running
SAFE	Core control during SAFE 0 main core_0 is frozen with clock gated 1 main core_0 is running
TEST	Core control during TEST 0 main core_0 is frozen with clock gated 1 main core_0 is running
RESET	Core control during RESET — main core_0 is always disabled during RESET.

62.3.2.29 Core Control Register (ME_CCTL2)

This register controls whether checker core_0s is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] = 1).

Figure 1520. Core Control Register (ME_CCTL2)

Address 0x1C8

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1418. ME_CCTL2 field descriptions

Field	Description
STOP0	Core control during STOP0 — checker core_0s is always disabled during STOP0.
HALT0	Core control during HALT0 — checker core_0s is always disabled during HALT0.
RUN3	Core control during RUN3 0 checker core_0s is disabled with clock gated 1 checker core_0s is running
RUN2	Core control during RUN2 0 checker core_0s is disabled with clock gated 1 checker core_0s is running
RUN1	Core control during RUN1 0 checker core_0s is disabled with clock gated 1 checker core_0s is running
RUN0	Core control during RUN0 0 checker core_0s is frozen with clock gated 1 checker core_0s is running
DRUN	Core control during DRUN 0 checker core_0s is frozen with clock gated 1 checker core_0s is running
SAFE	Core control during SAFE 0 checker core_0s is frozen with clock gated 1 checker core_0s is running
TEST	Core control during TEST 0 checker core_0s is frozen with clock gated 1 checker core_0s is running
RESET	Core control during RESET — checker core_0s is always disabled during RESET.

62.3.2.30 Core Control Register 3 (ME_CCTL3)

This register controls whether main core_1 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] = 1).

Figure 1521. Core Control Register 3 (ME_CCTL3)

Address 0x1CA

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1419. Core Control Register 3 (ME_CCTL3) Field Descriptions

Field	Description
STOP0	Core control during STOP0 — main core_1 is always disabled during STOP0 .
HALT0	Core control during HALT0 — main core_1 is always disabled during HALT0 .
RUN3	Core control during RUN3 0 main core_1 is disabled with clock gated 1 main core_1 is running
RUN2	Core control during RUN2 0 main core_1 is disabled with clock gated 1 main core_1 is running
RUN1	Core control during RUN1 0 main core_1 is disabled with clock gated 1 main core_1 is running
RUN0	Core control during RUN0 0 main core_1 is frozen with clock gated 1 main core_1 is running
DRUN	Core control during DRUN 0 main core_1 is frozen with clock gated 1 main core_1 is running
SAFE	Core control during SAFE 0 main core_1 is frozen with clock gated 1 main core_1 is running
TEST	Core control during TEST 0 main core_1 is frozen with clock gated 1 main core_1 is running
RESET	Core control during RESET — main core_1 is always disabled during RESET .

62.3.2.31 Core Control Register (ME_CCTL4)

This register controls whether the HSM is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] = 1).

Figure 1522. Core Control Register (ME_CCTL4)

Address 0x1CC

Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1420. ME_CCTL4 field descriptions

Field	Description
STOP0	Core control during STOP0 — the HSM is always disabled during STOP0.
HALT0	Core control during HALT0 — the HSM is always disabled during HALT0.
RUN3	Core control during RUN3 0 the HSM is disabled with clock gated 1 the HSM is running
RUN2	Core control during RUN2 0 the HSM is disabled with clock gated 1 the HSM is running
RUN1	Core control during RUN1 0 the HSM is disabled with clock gated 1 the HSM is running
RUN0	Core control during RUN0 0 the HSM is frozen with clock gated 1 the HSM is running
DRUN	Core control during DRUN 0 the HSM is frozen with clock gated 1 the HSM is running
SAFE	Core control during SAFE 0 the HSM is frozen with clock gated 1 the HSM is running
TEST	Core control during TEST 0 the HSM is frozen with clock gated 1 the HSM is running
RESET	Core control during RESET — the HSM is always disabled during RESET.

62.3.2.32 Core Address Register 0 (ME_CADDR0)

This register gives the boot address for peripheral core_2 and a bit for controlling whether peripheral core_2 is to be reset on the next mode change that has peripheral core_2 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while $ME_GS[S_MTRANS] = 1$). A write access to this register during this time will result in the $ME_IS[ICONF_CC]$ flag being asserted.

Note: The reset value of the $ME_CADDR0[ADDR]$ field is determined by the chip configuration and boot mode.

Figure 1523. Core Address Register 0 (ME_CADDR0)

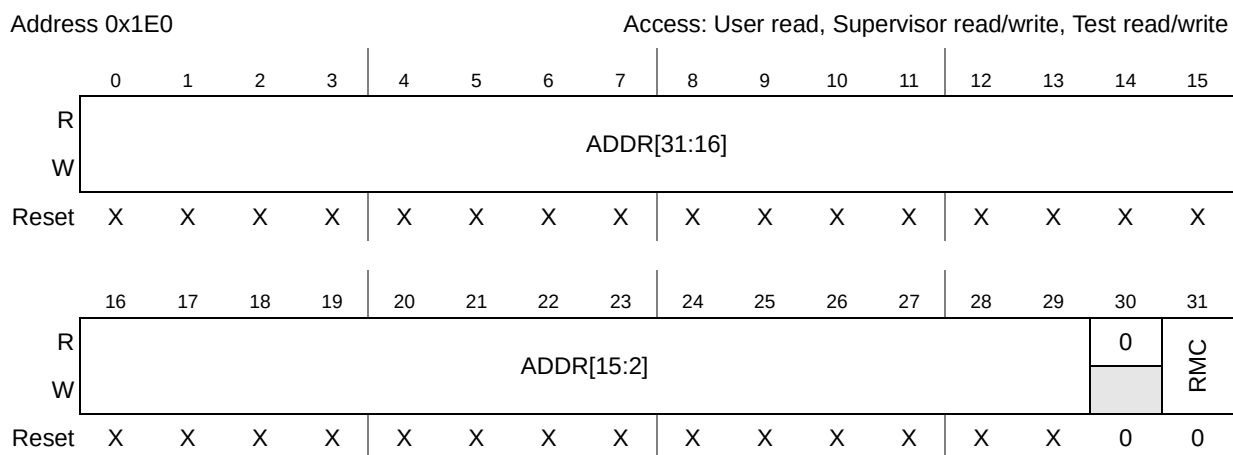


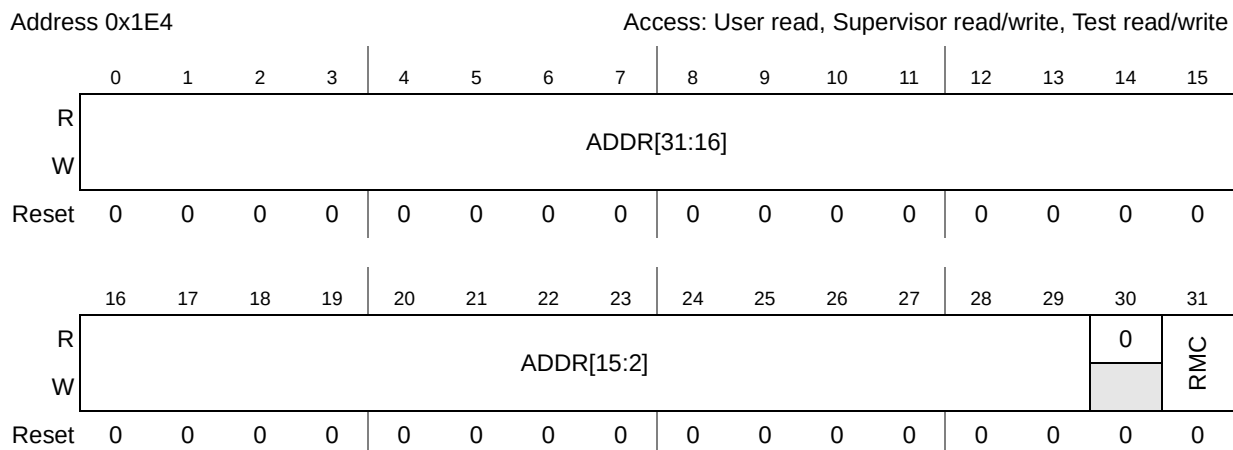
Table 1421. ME_CADDR0 field descriptions

Field	Description
ADDR	Core Address — This field is used by peripheral core_2 as the boot address (32-bit word aligned) when peripheral core_2 next exits reset.
RMC	Reset on Mode Change — peripheral core_2 will be reset on the next mode change that has peripheral core_2 configured to be running in the target mode. 0 peripheral core_2 will not be reset on the next mode change 1 peripheral core_2 will be reset on the next mode change that has peripheral core_2 configured to be running in the target mode

62.3.2.33 Core Address Register 1 (ME_CADDR1)

This register gives the boot address for main core_0 and a bit for controlling whether main core_0 is to be reset on the next mode change that has main core_0 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while $ME_GS[S_MTRANS] = 1$). A write access to this register during this time will result in the $ME_IS[ICONF_CC]$ flag being asserted.

Figure 1524. Core Address Register 1 (ME_CADDR1)**Table 1422. ME_CADDR1 field descriptions**

Field	Description
ADDR	Core Address — This field is used by main core_0 as the boot address (32-bit word aligned) when main core_0 next exits reset.
RMC	Reset on Mode Change — main core_0 will be reset on the next mode change that has main core_0 configured to be running in the target mode. 0 main core_0 will not be reset on the next mode change 1 main core_0 will be reset on the next mode change that has main core_0 configured to be running in the target mode

62.3.2.34 Core Address Register 3 (ME_CADDR3)

This register gives the boot address for main core_1 and a bit for controlling whether main core_1 is to be reset on the next mode change that has main core_1 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] = 1). A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted.

Figure 1525. Core Address Register 3 (ME_CADDR3)

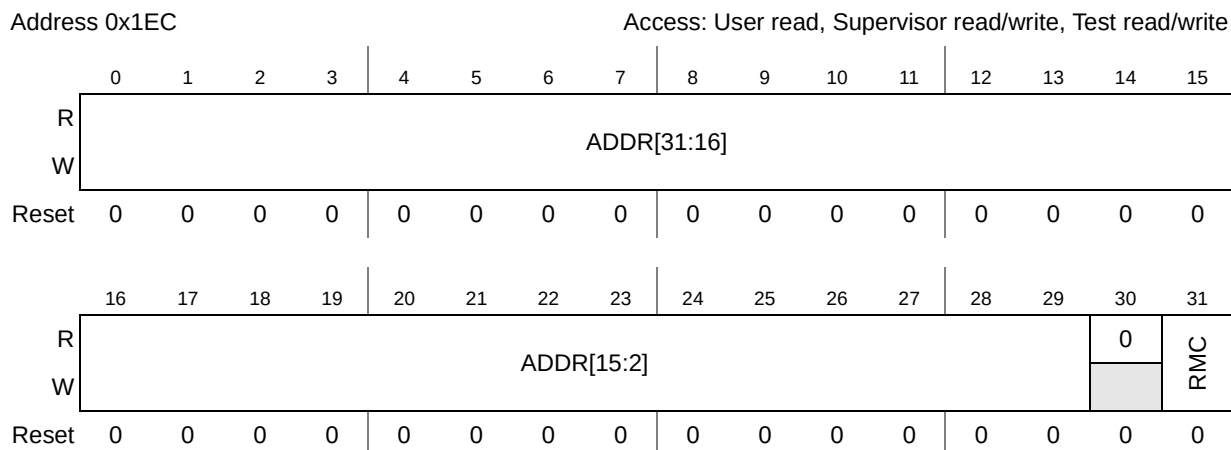


Table 1423. ME_CADDR3 field descriptions

Field	Description
ADDR	Core Address — This field is used by main core_1 as the boot address (32-bit word aligned) when main core_1 next exits reset.
RMC	Reset on Mode Change — main core_1 will be reset on the next mode change that has main core_1 configured to be running in the target mode. 0 main core_1 will not be reset on the next mode change 1 main core_1 will be reset on the next mode change that has main core_1 configured to be running in the target mode

62.3.2.35 Core Address Register 4 (ME_CADDR4)

This register gives the boot address for the HSM and a bit for controlling whether the HSM is to be reset on the next mode change that has the HSM configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] = 1). A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted.

Note: The reset value of the ADDR field of the ME_CADDR4 is determined by the chip configuration.

Figure 1526. Core Address Register 4 (ME_CADDR4)

Address 0x1F0				Access: User read, Supervisor read/write, Test read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	ADDR[31:16]																		
W																			
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	ADDR[15:2]															0	RMC		
W																			
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0			

Table 1424. ME_CADDR4 field descriptions

Field	Description
ADDR	Core Address — This field is used by the HSM as the boot address (32-bit word aligned) when main core_0 next exits reset.
RMC	Reset on Mode Change — The HSM will be reset on the next mode change that has the HSM configured to be running in the target mode. 0 the HSM will not be reset on the next mode change 1 the HSM will be reset on the next mode change that has the HSM configured to be running in the target mode

62.4 Functional description

62.4.1 Mode transition request

The transition from one mode to another mode is normally handled by software by accessing the mode control register ME_MCTL. But the in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<mode>_MC register. The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please see [Section 62.4.5, Mode Transition Interrupts](#).

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means

that transition requests such as RUN0...3 → RUN0...3, DRUN → DRUN, SAFE → SAFE, and TEST → TEST are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set till the status in the ME_GS register matches the configuration programmed in the respective ME_<mode>_MC register.

Note: It is recommended that software polls the S_MTRANS bit in the ME_GS register after requesting a transition to HALT0 or STOP0 modes.

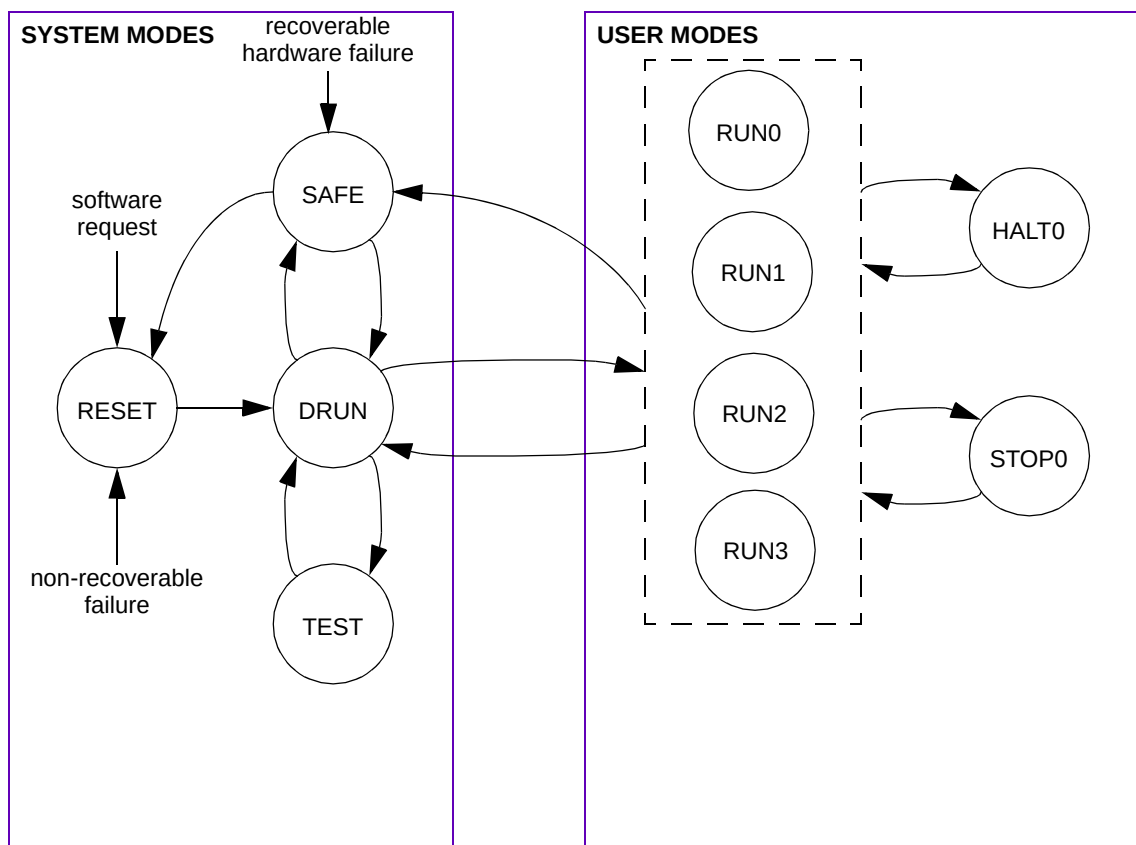


Figure 1527. MC_ME Mode Diagram

62.4.2 Modes details

62.4.2.1 RESET mode

The chip enters this mode on the following events:

- from SAFE, DRUN, RUN0...3, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with either "0000" for a 'functional' reset or "1111" for a 'destructive' reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the ME_RESET_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

62.4.2.2 DRUN mode

The chip enters this mode on the following events:

- automatically from RESET mode after completion of the reset sequence
- from RUN0...3, SAFE, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0011"

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz int. RC osc. as the system clock. The clock source configuration except system clock source 0 can be chosen by preloaded information in flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs

Note: .Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

62.4.2.3 SAFE mode

The chip enters this mode on the following events:

- from any mode except RESET when the TARGET_MODE bit field of the ME_MCTL register is written with "0010"
- from any mode except RESET due to a SAFE mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see [Chapter 57: Reset Generation Module \(MC_RGM\)](#), for details)

Note: If a hardware SAFE mode request occurs during RESET, depending on the timing of the SAFE mode request, SAFE mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after DRUN entry. The SAFE mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a SAFE mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

If the SAFE mode is requested by hardware, a SAFE mode interrupt is generated. See [Section 62.4.5.3, SAFE Mode Transition Interrupt](#) for details.

If the SAFE mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the SAFE mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

Note: If software requests to change to the SAFE mode and then requests to change back to the parent mode before the mode transition is completed, the chip's final mode after mode transition will be the SAFE mode, and an invalid mode transition interrupt will be generated.

As long as a SAFE event is active, the system remains in the SAFE mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
 - re-initialize the chip via the DRUN mode, or
 - completely reset the chip via the RESET mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the PDO bit of the ME_SAFE_MC register should be set. The input levels remain unchanged.

62.4.2.4 TEST mode

The chip enters this mode on the following events:

- from the DRUN mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0001”

As soon as any of the above events has occurred, a TEST mode transition request is generated. The mode configuration information for this mode is provided by the ME_TEST_MC register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the SYSCLK bit field to “1111”, and in this case, the only way to exit this mode is via a chip reset.

This mode is intended to be used by software

- to execute software test routines

Note: .Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

62.4.2.5 RUN0...3 modes

The chip enters one of these modes on the following events:

- from the DRUN, SAFE, or another RUN0...3 mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0100...0111”
- from the HALT0 mode due to an interrupt event
- from the STOP0 mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a RUN0...3 mode transition request is generated. The mode configuration information for these modes is provided by the ME_RUN0...3_MC registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

Note: .Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

62.4.2.6 HALT0 mode

The chip enters this mode on the following events:

- from one of the RUN0...3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1000”.

As soon as any of the above events has occurred, a HALT0 mode transition request is generated. The mode configuration information for this mode is provided by

ME_HALT0_MC register. This mode is quite configurable, and the ME_HALT0_MC register should be programmed according to the system needs. The flash can be put in low-power or power-down mode as needed. If there is a HALT0 mode request while an interrupt request is active, the transition to HALT0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the core clocks frozen the additional core clocks frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (i.e., within a few system clock cycles of an interrupt event)

The normal exit of the HALT0 mode is from an interrupt event. Wakeup events will not cause HALT0 mode exit.

Note: It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on HALT0 mode exit to ensure that the previous RUN0...3 mode configuration has been fully restored before executing critical code.

62.4.2.7 STOP0 mode

The chip enters this mode on the following events:

- from one of the RUN0...3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with "1010".

As soon as any of the above events has occurred, a STOP0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP0_MC register. This mode is fully configurable, and the ME_STOP0_MC register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- the primary PLL
- the secondary PLL

The flash can be put in power-down mode as needed. If there is a STOP0 mode request while any interrupt or wakeup event is active, the transition to STOP0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- the core clock frozen
- the additional core clocks frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started)

The normal exit from the STOP0 mode is by an interrupt or a wakeup event.

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the STOP0 mode, the 16 MHz internal RC oscillator clock is selected as the system clock until the target clock is available.

Note: .It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on STOP0 mode exit to ensure that the previous RUN0...3 mode configuration has been fully restored before executing critical code.

62.4.3 Mode transition process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

Since the MC_ME ensures that all resources are in the proper state during the mode transition process, the following, for example, do not need to be checked by software.

- When the XOSC is enabled in a mode configuration register, a mode transition to that mode will not complete until the End Of Count timeout occurs for the oscillator. Software does not need to interrogate the End Of Count timeout status bit.
- When a PLL is enabled in a mode configuration register, a mode transition to that mode will not complete until the PLL is locked. Software does not need to interrogate the PLL's lock status bit.

62.4.3.1 Target mode request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the TARGET_MODE bit field is not updated. An optional interrupt can be generated for invalid mode requests. See [Section 62.4.5, Mode Transition Interrupts](#), for details.

In the case of mode transitions occurring because of hardware events such as a reset, a SAFE mode request, or interrupt requests and wakeup events to exit from low-power modes, the TARGET_MODE bit field of the ME_MCTL register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit S_MTRANS of the ME_GS register.

A RESET mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The RESET mode request has the highest priority, and the MC_ME is kept in the RESET mode during the entire reset sequence.

The SAFE mode request has the next highest priority after reset. It can be generated either by software via the ME_MCTL register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

62.4.3.2 Target mode configuration loading

On completion of the [Target mode request](#) step, the target mode configuration from the ME_<target mode>_MC register is loaded to start the resources (voltage sources, clock sources, flash, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in [Table 1425](#). A '✓' indicates that a given resource is configurable for a given mode.

Table 1425. MC_ME resource control overview

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
IRC	on	√ on	on	on	on	on	on
XOSC	off	√ off	off	√ off	√ off	√ off	√ off
PLL0	off	√ off	off	√ off	√ off	√ off	off
PLL1	off	√ off	off	√ off	√ off	√ off	√ off
FLASH	normal	√ normal	normal	√ normal	√ normal	√ low-power	√ power-down
MVREG	on	on	on	on	on	on	on
PDO	off	√ off	√ on	off	off	off	√ off

62.4.3.3 Peripheral clocks disable

On completion of the [Target mode request](#) step and the system clock frequency ramp-down of the [System clock switching](#) step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0...7 and ME_LP_PC0...7, and the peripheral control registers ME_PCTLn

Note: *The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.*

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a SAFE mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The SAFE mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please see [Section 62.4.6, Peripheral Clock Gating](#), for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the SAFE mode.

62.4.3.4 Processor low-power mode entry

If, on completion of the [Peripheral clocks disable](#) step, the mode transition is to the HALT0 mode, the MC_ME requests the processor to enter its halted state. The processor acknowledges its halt state request after completing all outstanding bus transactions.

If, on completion of the [Peripheral clocks disable](#) step, the mode transition is to the STOP0 mode, the MC_ME requests the processor to enter its stopped state. The processor acknowledges its stop state request after completing all outstanding bus transactions.

62.4.3.5 Processor clocks disable

If, on completion of the [Processor low-power mode entry](#) step, and all processors which are configured in the ME_CCTLn registers to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processors which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes.

Caution: Clocks to the whole chip including the processor and system memory can be disabled in TEST mode.

62.4.3.6 System clock disable

If, on completion of the [Processor clocks disable](#) step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

62.4.3.7 Clock sources switch-on

On completion of the [Processor low-power mode entry](#) step, the MC_ME switches on all clock sources based on the <clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

- the 16 MHz internal RC oscillator
- the external crystal oscillator
- the primary PLL
- the secondary PLL

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the S_<clock source> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

62.4.3.8 Flash memory module switch-on

On completion of the step, if the flash memory needs to be switched to normal mode from its low-power or power-down mode based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, the MC_ME requests the flash memory to exit from its low-power/power-down mode. When the flash memory is

available for access, the S_FLA bit field of the ME_GS register is updated to “11” by hardware.

Caution: It is illegal to switch the FLASH or the DFLASH from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

62.4.3.9 Pad outputs-on

On completion of the step, if the PDO bit of the ME_<target mode>_MC register is cleared, then

- all pad outputs are enabled to return to their previous state
- slew rate control mechanism is switched on

62.4.3.10 Peripheral clocks enable

Based on the current and target chip modes, the peripheral configuration registers ME_RUN_PC0...7, ME_LP_PC0...7, and the peripheral control registers ME_PCTLn, the MC_ME enables the clocks for selected modules as required. If the value of the PWRLVL field of the ME_<target mode>_MC register is less than that of the ME_<current mode>_MC register's, this step is executed only after the [System clock switching](#) process is completed.

62.4.3.11 System and processor clocks enable

On completion of the [Flash memory module switch-on](#) step, the MC_ME enables the non-processor system clock and the clocks of all processors which are configured in the ME_CCTLn registers to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of all processors which are configured in the ME_CCTLn registers to be running in the target mode and which have their RMC bits in the ME_CADDRn registers set to '1'. The core_rst_b outputs remain asserted until the end of the mode transition as indicated by the S_MTRANS bit of the MW_GS register at which time they are deasserted.

If the value of the PWRLVL field of the ME_<target mode>_MC register is different from that of the ME_<current mode>_MC register's, the mode change handshake will not be initiated until after the system clock frequency ramp-down step of the [System clock switching](#) process is completed.

62.4.3.12 Processor low-power mode exit

On completion of the [System and processor clocks enable](#) step, the MC_ME requests all processors which are configured in the ME_CCTLn registers to be disabled in the current mode and to be running in the target mode by deasserting the corresponding core_stop bits.

62.4.3.13 System clock switching

Based on the SYSCLK bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz int. RC osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the external crystal osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the primary PLL (PHI) takes effect only after the S_XOSC bit of the ME_GS register is set by hardware (i.e., the external crystal oscillator has stabilized).
- The target clock configuration for the secondary PLL takes effect only after the S_PLL0 bit of the ME_GS register is set by hardware (i.e., the primary PLL has stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with "1111". This is possible only in the TEST mode.

If the value of the PWRLVL field of the ME_<target mode>_MC register not equal to that of the ME_<current mode>_MC register's, a progressive clock switching is performed in the

The current system clock configuration can be observed by reading the S_SYSCLK bit field of the ME_GS register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the *Clock sources switch-on* process has completed if the target system clock source is one of the following:
 - the 16 MHz internal RC oscillator
 - the external crystal oscillator
 - the primary PLL
- the *Peripheral clocks disable* process has completed in order not to change the system clock frequency before peripherals close their internal activities
- the *Peripheral clocks enable* process has completed

An overview of system clock source selection possibilities for each mode is shown in [Table 1426](#). A '✓' indicates that a given clock source is selectable for a given mode.

Table 1426. MC_ME System Clock Selection Overview

System Clock Source	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
16 MHz int. RC osc.	✓ (default)	✓ (default)	✓ (default)	✓ (default)	✓ (default)	✓ (default)	✓ (default)
external crystal osc.		✓		✓	✓	✓	✓

Table 1426. MC_ME System Clock Selection Overview (continued)

System Clock Source	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
primary PLL (PHI)		√		√	√	√	√
secondary PLL		√		√	√	√	
system clock is disabled		√ ⁽¹⁾					

1. disabling the system clock during TEST mode will require a reset in order to exit TEST mode

62.4.3.14 Pad switch-off

If the PDO bit of the ME_<target mode>_MC register is '1' then

- the outputs of the pads are forced to the high impedance state if the target mode is SAFE or TEST

This step is executed only after the [Peripheral clocks disable](#) process has completed.

62.4.3.15 Clock sources switch-off

Based on the chip mode and the <clock source>ON bits of the ME_<mode>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<clock source> of the ME_GS register to '0'. The following clock sources switched off at this step:

- the 16 MHz internal RC oscillator
- the external crystal oscillator
- the primary PLL
- the secondary PLL

For the clock sources related to the system clock, this step is executed only after the [System clock switching](#) process has completed.

62.4.3.16 Flash memory switch-off

Based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the flash is to be put in its low-power or power-down mode, the MC_ME requests the flash to enter the corresponding power mode and waits for the flash to acknowledge. The exact power mode status of the flash is updated in the S_FLA bit field of the ME_GS register. This step is executed only when the [Processor clocks disable](#) and [System clock disable](#) processes have completed.

62.4.3.17 Current mode update

The current mode status bit field S_CURRENT_MODE of the ME_GS register is updated with the target mode bit field TARGET_MODE of the ME_MCTL register when :

- all the updated status bits in the ME_GS register match the configuration specified in the ME_<target mode>_MC register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

Note: *SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the ME_GS.S_RC bit to be set. This is to ensure that the SAFE mode is entered as quickly as possible.*

Software can monitor the mode transition status by reading the S_MTRANS bit of the ME_GS register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the ME_DMTS register can indicate which process is still in progress.

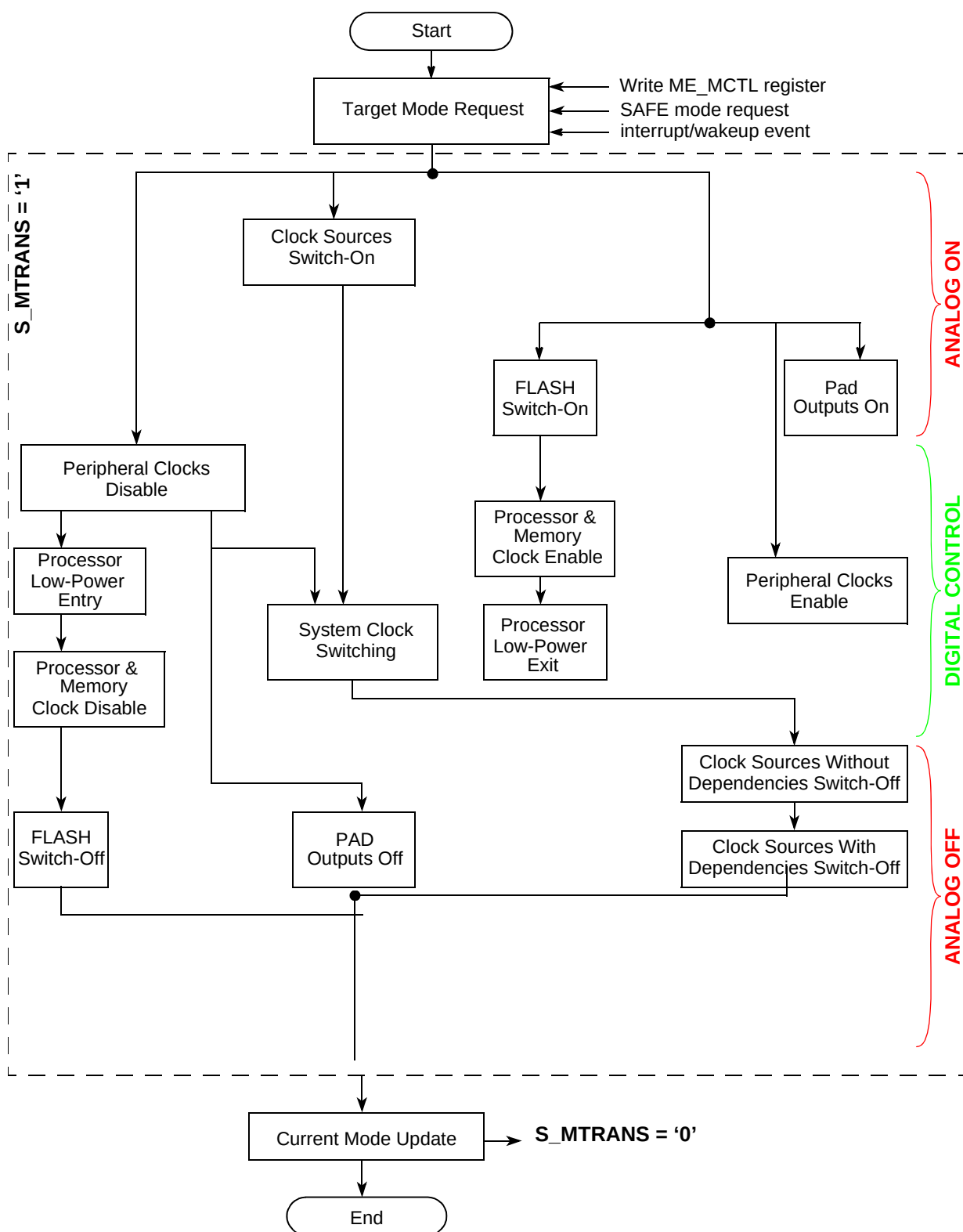


Figure 1528. MC_ME Transition Diagram

62.4.4 Protection of Mode Configuration Registers

While programming the mode configuration registers ME_<mode>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the 16 MHz int. RC osc. is selected as the system clock, IRC must be on.
- If the external crystal osc. clock is selected as the system clock, XOSC must be on.
- If the primary PLL (PHI) clock is selected as the system clock, PLL0 must be on.
- If the secondary PLL clock is selected as the system clock, PLL1 must be on.

Note: .Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration “00” for the FLAON bit field is reserved.
- Configuration “00” for the DFLAON bit field is reserved.
- System clock configurations marked as ‘reserved’ may not be selected.
- Configuration “1111” for the SYSCLK bit field is allowed only for the TEST mode, and only in this case may all system clock sources be turned off.

Caution: If the system clock is stopped during *TEST* mode, the chip can exit only via a system reset.

62.4.5 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a SAFE mode transition not due to a software request, and indicating when a mode transition has completed.

62.4.5.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the ME_<mode>_MC registers violating the protection rules mentioned in the [Section 62.4.4, Protection of Mode Configuration Registers](#), the interrupt pending bit I_ICONF of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF of the ME_IM register is ‘1’.

In addition, during a mode transition, if a clock source has been configured in the ME_<target mode>_MC register to be off and a peripheral requiring this clock source to be on has been enabled via the ME_RUN_PC0...7/ME_LP_PC0...7 and ME_PCTLn registers, the interrupt pending bit I_ICONF_CU of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CU of the ME_IM register is ‘1’.

If an attempt to write to one of the ME_CADDRn registers is made after a mode change request has been made via the second write to the ME_CTL register and before the completion of that mode transition as indicated by the S_MTRANS bit in the ME_GS register deasserting, the interrupt pending bit I_ICONF_CC of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CC of the ME_IM register is ‘1’.

62.4.5.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the SAFE mode and the SAFE mode request from MC_RGM is active, and if the target mode requested is other than RESET or SAFE, then this new

mode request is considered to be invalid, and the S_SEA bit of the ME_IMTS register is set.

- If the TARGET_MODE bit field of the ME_MCTL register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the S_NMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If some of the chip modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the S_DMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit S_MRI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition to a mode other than SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is '1'), the mode transition illegal status bit S_MTI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a new mode request occurs while a mode transition to the SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is '1'), the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a HALT0 mode request occurs while an interrupt request is active, the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a STOP0 mode request occurs while a wakeup or interrupt request is active, the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.

Note: As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA, S_MRI, and S_MTI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the RESET or SAFE mode irrespective of the mode transition status.
- As the exit of HALT0 and STOP0 modes depends on the interrupts of the system which can occur at any instant, these requests to return to RUN0...3 modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined 'reasonable' amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.

Warning: Depending on the cause of the mode transition not completing, aborting via a request back to the parent mode may not be able to recover. In this case, software should request a change to either SAFE or RESET mode, since the cause of the hang-up is most likely due to a hardware fault.

- Self-transition requests (e.g., RUN0 → RUN0) are not considered as invalid even when the mode transition process is active (i.e., S_MTRANS is '1'). During the low-power mode exit process, if the system is not able to enter the respective RUN0...3 mode properly (i.e., all status bits of the ME_GS register match with configuration bits in the ME_<mode>_MC register), then software can only request the SAFE or RESET mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit I_IMODE of the ME_IS register is set, and an interrupt request is generated if the mask bit M_IMODE of the ME_IM register is '1'.

62.4.5.3 SAFE Mode Transition Interrupt

Whenever the system enters the SAFE mode as a result of a SAFE mode request from the MC_RGM due to a hardware failure, the interrupt pending bit I_SAFE of the ME_IS register is set, and an interrupt is generated if the mask bit M_SAFE of ME_IM register is '1'.

The SAFE mode interrupt pending bit can be cleared only when the SAFE mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a SAFE mode request). If the system is already in SAFE mode, any new SAFE mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the SAFE mode interrupt pending bit is not set when the SAFE mode is entered by a software request (i.e., programming of ME_MCTL register).

62.4.5.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (i.e., the S_MTRANS bit of ME_GS register transits from '1' to '0'), the interrupt pending bit I_MTC of the ME_IS register is set, and an interrupt request is generated if the mask bit M_MTC of the ME_IM register is '1'. The interrupt bit I_MTC is not set when entering low-power modes HALT0 and STOP0 in order to avoid the same event requesting the immediate exit of these low-power modes.

62.4.6 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0...7 are chosen only during the software running modes DRUN, TEST, SAFE, and RUN0...3. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTLn registers.

The low-power peripheral configuration registers ME_LP_PC0...7 are chosen only during the low-power modes HALT0 and STOP0. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit

which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTLn registers.

Any modifications to the ME_RUN_PC0...7, ME_LP_PC0...7, and ME_PCTLn registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the processor enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTLn register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits. Any further modifications of the ME_RUN_PC0...7, ME_LP_PC0...7, and ME_PCTLn registers during a debug session will take affect immediately without requiring any new mode request.

62.4.7 Application Example

[Figure 1529](#) shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

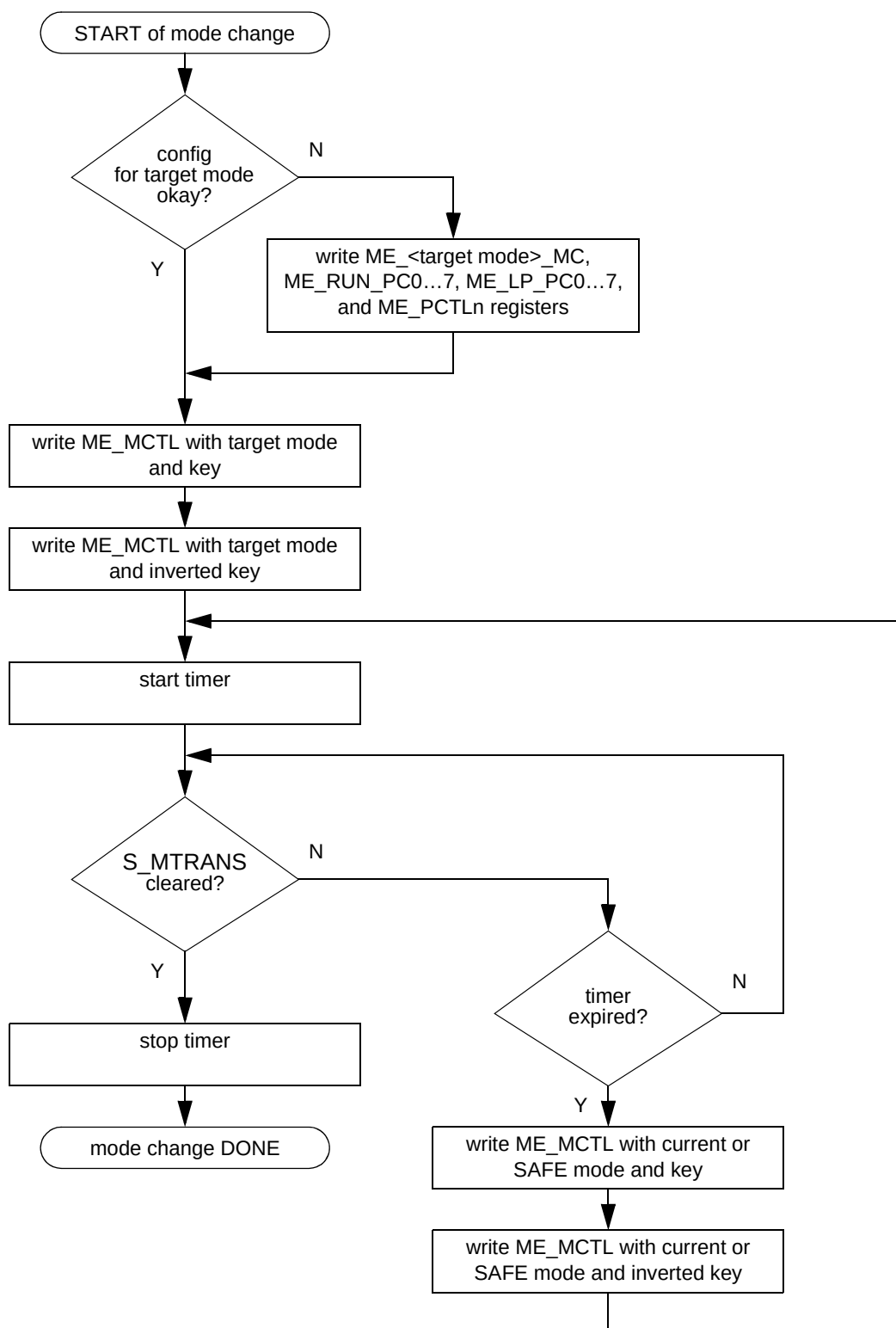


Figure 1529. MC_ME Application Example Flow Diagram

63 e200z420n3 Core Debug Support

This chapter describes the debug features of the e200z420n3 core.

63.1 Overview

Internal debug support in the e200z420n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware-based debugger, which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *PowerISA 2.06*. e200z420n3 supports a subset of these defined facilities. In addition to the facilities defined in *PowerISA 2.06*, e200z420n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

63.1.1 Software debug facilities

e200z420n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors that are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 (DBCR0_{IDM}). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When DBCR0_{IDM} is cleared (and EDBCR0_{EDM} is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when DBCR0_{IDM} is cleared (or is overridden by EDBCR0_{EDM} being set and EDBRAC0 indicating no resource is “owned” by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

63.1.1.1 <asisitalic>PowerISA 2.06 compatibility

The e200z420n3 core implements a subset of the <asisitalic>PowerISA 2.06 internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

63.1.2 Additional debug facilities

In addition to the debug functionality defined in <asisitalic>PowerISA 2.06, e200z420n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See [Section 63.1.4, Sharing debug resources by software/hardware](#).) A data trace port is also provided.

63.1.2.1 Data trace port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z420n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the “Data Trace Control Register (DTC)” section in the Core (e200z420n3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

63.1.3 Hardware debug facilities

The e200z420n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z420n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug mode enable bit in External Debug Control register 0 (EDBCR0_{EDM}), which is also aliased to DBCR0_{EDM}. Setting EDBCR0_{EDM} overrides the Internal Debug mode enable bit DBCR0_{IDM} unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the “hardware-owned” debug facilities. In addition, since resources are “owned” by the hardware debugger, inconsistent values may be present if software attempts to read “hardware-owned” debug-related resources.

When hardware debug is enabled by setting EDBCR0_{EDM}=1, the control registers and resources described in [Section 63.3, Debug registers](#) are reserved for use by the external debugger. The same events described in [Section 63.2, Software debug events and exceptions](#) are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no

debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. EDBCR0_{EDM}, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

63.1.4 Sharing debug resources by software/hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When EDBCR0_{EDM} is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources that set DBSR bits when EBCR0_{IDM}=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources that set EDBSR0 bits when EDBCR0_{EDM}=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, (EDBCR0_{EDM}=1 and EDBRAC0_{IDM}=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

63.1.4.1 Simultaneous hardware and software debug event handling

Since it is possible that a “hardware-owned” resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented that guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

[Figure 1530](#) shows the e200z420n3 debug resources.

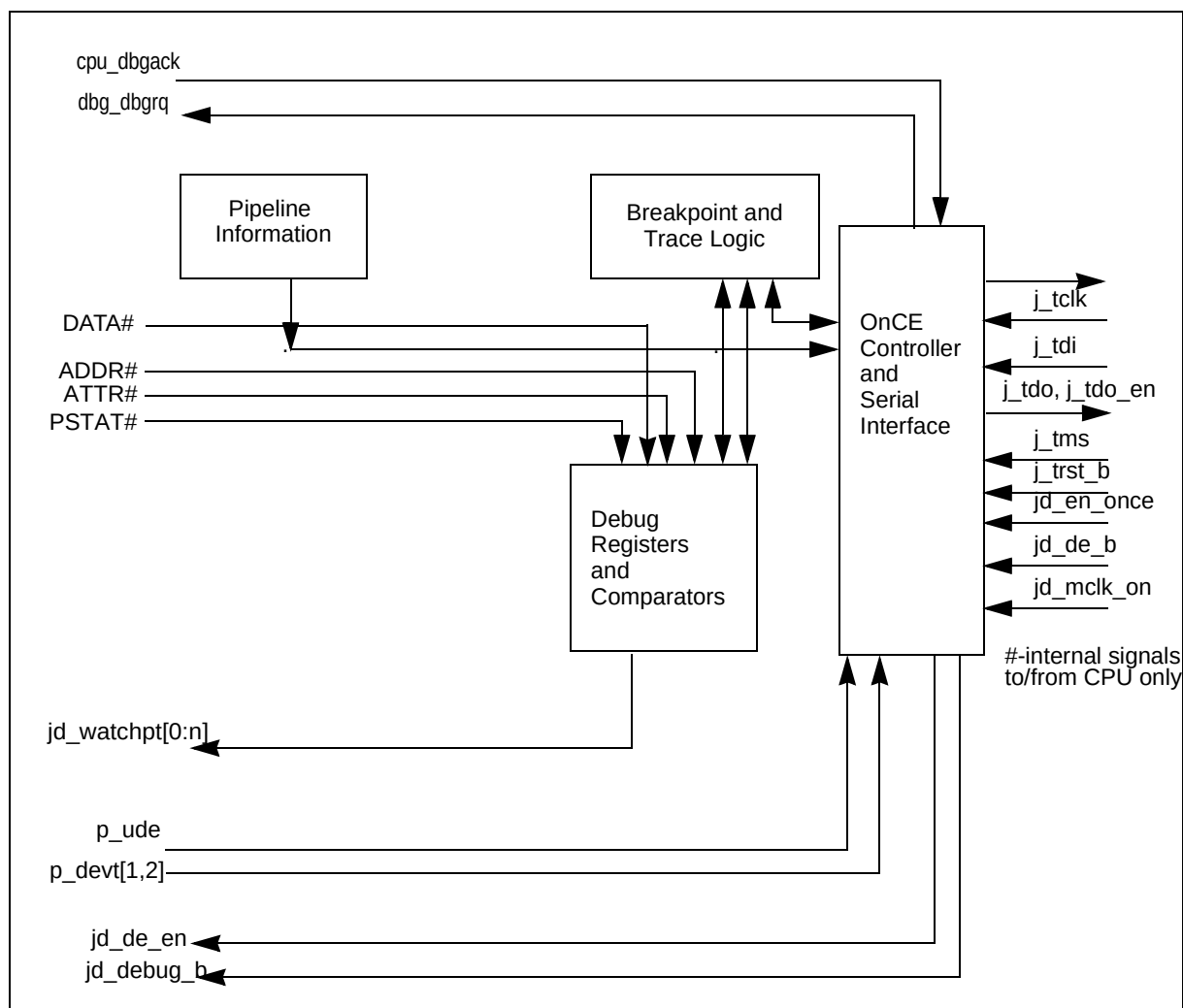


Figure 1530. e200z420n3 debug resources

63.2 Software debug events and exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}=1$) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in $EDBRAC0$). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers ($DBCR0-8$). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource that is “owned” by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE} , a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when $MSR_{DE}=0$ and $DBCRO_{IDM}=1$. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE} . A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while $MSR_{DE}=0$, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction that enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** that causes both MSR_{DE} and $DBCRO_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *PowerISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z420n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2, which are described in [Section 63.2.11, External debug event](#).
- The Performance Monitor Interrupt event PMI, which is described in [Section 63.2.13, Performance Monitor Interrupt debug event](#).

The e200z420n3 debug configuration supports most of these event types. Unsupported *PowerISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

63.2.1 Instruction Address Compare event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

63.2.2 Data Address Compare event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the DBSR_{DAC{R,W}} bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the DBSR_{DAC{R,W}} bits are set.

Note: In contrast to the *PowerISA 2.06* definition, Data Address Compare events on e200z420n3 do not prevent the load or store class instruction from completing. If a load or store class instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting DBSR_{IDE}) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the DBSR_{DAC_OFST} field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the DBSR_{IDE} bit will be set. In addition to occurring when DBCR0_{IDM}=1, this circumstance can also occur when EDBCR0_{EDM}=1 and the event is hardware-owned, in which case EDBSR0_{IDE} will be set.

Note: DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note: DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, i.e. no data comparisons are performed since these instructions do not have associated data values.

Note: DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note: DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.

Note: DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note: Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

63.2.2.1 Data Address Compare event status updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the DBSR_{DAC_OFST} field when the DVC DAC status is recorded.

[Table 1427](#) outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in [Table 1427](#), the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 1427. DAC events and resultant updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.

Table 1427. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
DVC DACx	DVC DACy, Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
DVC DACx	DVC DACy, Normal Ldst	Ldst instruction, no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
DVC DACx	DVC DACy, Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i> <i>Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
DVC DACx	DVC DACy, Normal Ldst	DACy, or DVC DACy, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1427. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DVC DACy, Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
DVC DACx	Any instruction (no exception)	DSI, with or without DAC, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. <i>Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
DVC DACx	Any instruction (no exception)	DACy, or DVC DACy Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1428 through Table 1431 show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 1428. DAC events and resultant updates, dual-issue case 1

Instruction sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)

Table 1428. DAC events and resultant updates, dual-issue case 1 (continued)

Instruction sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.</i> <i>Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.</i>

Table 1429. DAC events and resultant updates, dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1429. DAC events and resultant updates, dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i> <i>Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). <i>Note: In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1430. DAC events and resultant updates, dual-issue case 3

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.

Table 1430. DAC events and resultant updates, dual-issue case 3 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1431. DAC events and resultant updates, dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.

Table 1431. DAC events and resultant updates, dual-issue case 4 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). <i>Note: in this case if $x=y$, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). <i>Note: in this case if $x=y$, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

63.2.3 Linked Instruction Address Compare and Data Address Compare events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note: Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

63.2.4 Trap debug event

A Trap debug event (TRAP) occurs if Trap debug events are enabled (DBCR0_{TRAP}=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the DBSR_{TRAP} bit is set to 1 to record the debug exception.

63.2.5 Branch Taken debug event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled (DBCR0_{BRT}=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus DBSR_{ID}E can not be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the DBSR_{BRT} bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

63.2.6 Instruction Complete debug event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled (DBCR0_{ICMP}=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The **se_sc** instruction does not fall into the category of an instruction whose execution is suppressed, since the

instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in $DSRR0$.

Instruction Complete debug events are not recognized if $MSR_{DE}=0$ at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed ($SRR0$ points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with $SRR0$ pointing to the instruction following the instruction that generated the EFPU Round exception, and $DSRR0$ will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}=1$, this circumstance can also occur when $EDBCR0_{EDM}=1$ and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}=1$, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while $MSR_{DE}=1$.

63.2.7 Interrupt Taken debug event

An Interrupt Taken debug event ($IRPT$) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}=1$) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using $SRR0/1$) cause an Interrupt Taken debug event. This event can occur and be recorded in $DBSR$ regardless of the setting of MSR_{DE} . When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in $DSRR0$ will be the address of the non-critical interrupt handler.

63.2.8 Critical Interrupt Taken debug event

A Critical Interrupt Taken debug event ($CIRPT$) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}=1$) and a critical interrupt occurs. Only critical class interrupts (an interrupt using $CSRR0/1$) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in $DBSR$ regardless of the setting of MSR_{DE} . When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in $DSRR0$ will be the address of the critical interrupt handler.

63.2.9 Return debug event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}=1$) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in $DBSR$ regardless of the setting of MSR_{DE} . When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If $MSR_{DE}=0$ at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If $MSR_{DE}=1$ at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

63.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}=1$) and an attempt is made to execute an **se_rfc**i instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If $MSR_{DE}=0$ at the time of the execution of the **se_rfc**i (i.e. before the MSR is updated by the **se_rfc**i), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If $MSR_{DE}=1$ at the time of the execution of the **se_rfc**i, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfc**i instruction.

63.2.11 External debug event

An External debug event ($DEVT1$, $DEVT2$) occurs if External debug events are enabled ($DBCR0_{DEVT1}=1$ or $DBCR0_{DEVT2}=1$), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

63.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in $DBCR0$. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

63.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}=1$), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

63.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register $EDBRAC0$, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in $EDBRAC0$ corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause

modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not “owned” by software is not guaranteed to be consistent if accessed (read) by software with a **mfsprr** instruction, other than the DBCR0_{EDM} bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

63.3.1 Debug Address and Value registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in [Figure 1531](#)) corresponds to the byte data value transferred to/from memory byte offset 0, 8, ..., and the least significant byte of the register (labeled B7 in [Figure 1531](#)) corresponds to byte offset 7, F, When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison; they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z420n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfsprr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

Figure 1531. DVC1, DVC2 registers

B0								B1								B2								B3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
B4								B5								B6								B7							
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

SPR 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset: Unaffected

63.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z420n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

63.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z420n3 adds some implementation specific bits to this register, as seen in [Figure 1532](#).

Figure 1532. Debug Control Register 0 (DBCR0) register

EDM	IDM	RST		ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1		DAC2		RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	0		CIRPT	CRET	0					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	

SPR - 308; Read/Write; Reset⁽¹⁾ - 0x0

1. DBCR0_{EDM} is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if DBCR0_{EDM}=0, as well as unconditionally by **m_por**. If DBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the DBCR0_{RST} field will be reset by **p_reset_b**. The DBCR0_{RST} field will always be reset by **p_reset_b** regardless of the value of DBCR0_{EDM}.

[Table 1432](#) provides bit definitions for Debug Control Register 0.

Table 1432. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRO–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If $MSR_{DE}=1$, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 p_dbrstc[1] pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 p_dbrstc[0] pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled

Table 1432. DBCR0 field descriptions (continued)

Bit	Name	Description
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1 IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled

Table 1432. DBCR0 field descriptions (continued)

Bit	Name	Description
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

63.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in [Figure 1533](#).

Figure 1533. DBCR1 register

IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	0	IAC3US	IAC3ER	IAC4US	IAC4ER	IAC34M	0
--------	--------	--------	--------	--------	---	--------	--------	--------	--------	--------	---

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 309; Read/Write; Reset⁽¹⁾ - 0x0

- Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1433](#) provides bit definitions for Debug Control Register 1.

Table 1433. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR _{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR _{PR} =1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC1 debug events are based on effective address and can only occur if MSR _{IS} =1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR _{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR _{PR} =1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC2 debug events are based on effective address and can only occur if MSR _{IS} =1

Table 1433. DBCR1 field descriptions (continued)

Bit	Name	Description
8:9	IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

63.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation. The DBCR2 register is shown in [Figure 1534](#).

Figure 1534. DBCR2 register

DAC1US	DAC1ER	DAC2US	DAC2ER	DAC12M	DAC1LNK	DAC2LNK	DVC1M	DVC2M	DVC1BE					DVC2BE																	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 310: Read/Write: Reset⁽¹⁾ - 0x0

- Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1434](#) provides bit definitions for Debug Control Register 2.

Table 1434. DBCR2 field descriptions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR _{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR _{PR} =1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR _{DS} =0 11 DAC1 debug events are based on effective address and can only occur if MSR _{DS} =1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR _{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR _{PR} =1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR _{DS} =0 11 DAC2 debug events are based on effective address and can only occur if MSR _{DS} =1

Table 1434. DBCR2 field descriptions (continued)

Bit	Name	Description
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies <i>Exact Address Compare</i> since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR.

Table 1434. DBCR2 field descriptions (continued)

Bit	Name	Description
12:13	DVC1M	Data Value Compare 1 Mode When DBCR4_{DVC1C}=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. <i>Note: Inactive byte lanes of the memory access are automatically masked.</i> When DBCR4_{DVC1C}=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved <i>Note: Inactive byte lanes of the memory access are automatically masked.</i>
14:15	DVC2M	Data Value Compare 2 Mode When DBCR4_{DVC2C}=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. <i>Note: Inactive byte lanes of the memory access are automatically masked.</i> When DBCR4_{DVC2C}=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved <i>Note: Inactive byte lanes of the memory access are automatically masked.</i>

Table 1434. DBCR2 field descriptions (continued)

Bit	Name	Description
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

63.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in [Figure 1535](#).

Figure 1535. DBCR4 register

0	DVC1C	0	DVC2C	0								0	DAC1XMH	0	DAC2XMH	DAC1XM				DAC2XM				DAC1CFG				DAC2CFG			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 563; Read/Write; Reset⁽¹⁾ - 0x0

1. DBCR4 is reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1435](#) provides bit definitions for Debug Control Register 4.

Table 1435. DBCR4 field descriptions

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.

Table 1435. DBCR4 field descriptions (continued)

Bit	Name	Description
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}=00$ 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}=00$ 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}=00$ 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). <i>Note: Unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}=0$; when $DBCR0_{DAC1} \neq 0$, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.</i>
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}=00$ 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}=00$ 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}=00$ 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. <i>Note: Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}=00$. When $DBCR0_{DAC2} \neq 0$, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.</i>

63.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in [Figure 1536](#)

Figure 1536. DBCR5 register

IAC5US	IAC5ER	IAC6US	IAC6ER	IAC56M	0	IAC7US	IAC7ER	IAC8US	IAC8ER	IAC78M	0
--------	--------	--------	--------	--------	---	--------	--------	--------	--------	--------	---

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 564; Read/Write; Reset⁽¹⁾ - 0x0

1. Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1433 provides bit definitions for Debug Control Register 5.

Table 1436. DBCR5 field descriptions

Bit	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR _{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR _{PR} =1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC5 debug events are based on effective address and can only occur if MSR _{IS} =1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR _{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR _{PR} =1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC6 debug events are based on effective address and can only occur if MSR _{IS} =1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved

Table 1436. DBCR5 field descriptions (continued)

Bit	Name	Description
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR _{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR _{PR} =1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC7 debug events are based on effective address and can only occur if MSR _{IS} =1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR _{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR _{PR} =0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR _{PR} =1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC8 debug events are based on effective address and can only occur if MSR _{IS} =1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

63.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in [Figure 1537](#).

Figure 1537. DBCR6 register

IAC1XM	IAC2XM	IAC3XM	IAC4XM	IAC5XM	IAC6XM	IAC7XM	IAC8XM
--------	--------	--------	--------	--------	--------	--------	--------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 603; Read/Write; Reset⁽¹⁾ - 0x0

1. DBCR6 is reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1437 provides bit definitions for Debug Control Register 6.

Table 1437. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 = Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Control. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

Table 1437. DBCR6 field descriptions (continued)

Bit	Name	Description
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101 – 1111 = Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

63.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in [Figure 1538](#).

Figure 1538. DBCR7 register

0	0	DAC3XMH	0	DAC4XMH	DAC3XM	DAC4XM	DAC3CFG	DAC4CFG
---	---	---------	---	---------	--------	--------	---------	---------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 596; Read/Write; Reset⁽¹⁾ - 0x0

1. DBCR7 is reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1438](#) provides bit definitions for Debug Control Register 7.

Table 1438. DBCR7 field descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High. DAC3XMH extends the range of the DAC3XM field. 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits

Table 1438. DBCR7 field descriptions (continued)

Bit	Name	Description
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001–1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

Table 1439. DBCR8 field descriptions (continued)

Bit	Name	Description
8:9	DAC34M	Data Address Compare 3/4 Mode 00 Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 No effect 1 DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 No effect 1 DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

63.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr DBSR,RS**. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug mode is enabled ($DBCR0_{IDM}=1$). When debug interrupts are enabled ($MSR_{DE}=1$, $DBCR0_{IDM}=1$ and $EDBCR0_{EDM}=0$, or $MSR_{DE}=1$, $DBCR0_{IDM}=1$, $EDBCR0_{EDM}=1$ and software is allocated resource(s) via $EDBRAC0$), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}=1$ and $EDBRAC0_{IDM}=1$), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in [Figure 1540](#).

Figure 1540. Debug Status Register (DBSR)

IDE	UDE	MRR	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST	0								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 304; Read/Clear; Reset - 0x1000 0000

Table 1440 provides bit definitions for the Debug Status Register.

Table 1440. DBSR field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $MSR_{DE}=0$ and $DBCRO_{IDM}=1$ and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store that is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 No reset occurred since these bits were last cleared by software 01 A hard reset occurred since these bits were last cleared by software 10 Reserved 11 Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred

Table 1440. DBSR field descriptions (continued)

Bit	Name	Description
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UD}[1]=1$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

63.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in [Figure 1541](#).

Figure 1541. Debug Data Effective Address Register (DDEAR)

Data Effective Address																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 600; Read/Write; Reset - unaffected

63.3.3 External Debug Resource Allocation Control (EDBRAC0) register

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}=1$. EDBRAC0 settings are ignored when $EDBCR0_{EDM}=0$.

Hardware-owned resources that generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources that generate debug events if $DBCR0_{IDM}=1$ update DBSR, causing debug interrupts to occur if $MSR_{DE}=1$. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug mode is enabled. When debug interrupts are enabled ($MSR_{DE}=1$, $DBCR0_{IDM}=1$ and $EDBCR0_{EDM}=0$, or $MSR_{DE}=1$, $DBCR0_{IDM}=1$ and $EDBCR0_{EDM}=1$ and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event that is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug mode is enabled ($EDBCR0_{EDM}=1$). When $EDBCR0_{EDM}=1$, a set bit in EDBSR0 by an event that is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}=1$, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in $DBCR0-8$ or $MPU0CSR0$ are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in [Figure 1542](#).

Figure 1542. External Debug Resource Allocation Control (EDBRAC0) register

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	DNI	DQM	0		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b**, reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

[Table 1440](#) provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 1441. EDBRAC0 field descriptions

Bit	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control. When EDBRAC0 _{IDM} =1, software writes to hardware-owned bits in DBCR0–8 via mtspr are ignored. 0 Internal Debug mode may not be enabled by software. DBCR0 _{IDM} is owned exclusively by hardware. mtspr DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 Internal Debug mode may be enabled by software. DBCR0 _{IDM} is owned by software. DBCR0 _{IDM} is software readable/writable.
2	RST	Reset Field Control 0 DBCR0 _{RST} owned exclusively by hardware debug. No mtspr access by software to DBCR0 _{RST} field. 1 DBCR0 _{RST} accessible by software debug. DBCR0 _{RST} is software readable/writable.
3	UDE	Unconditional Debug Event 0 Event owned by hardware debug. 1 Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{ICMP} field. 1 Event owned by software debug. DBCR0 _{ICMP} is software readable/writable.
5	BRT	Branch Taken Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{BRT} field. 1 Event owned by software debug. DBCR0 _{BRT} is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{IRPT} field. 1 Event owned by software debug. DBCR0 _{IRPT} is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{TRAP} field. 1 Event owned by software debug. DBCR0 _{TRAP} is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC1 control and status fields. 1 Event owned by software debug. IAC1 control fields are software readable/writable.

Table 1441. EDBRAC0 field descriptions (continued)

Bit	Name	Description
9	IAC2	Instruction Address Compare 2 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC2 control and status fields. 1 Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC3 control and status fields. 1 Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC4 control and status fields. 1 Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 Event owned by hardware debug. No mtspr access by software to DAC1 control and status fields. 1 Event owned by software debug. DAC1 control fields are software readable/writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 Events owned by hardware debug. No mtspr access by software to DAC3 and DAC4 control and status fields. 1 Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 Event owned by hardware debug. No mtspr access by software to DAC2 control and status fields. 1 Event owned by software debug. DAC2 control fields are software readable/writable.
15	—	Reserved
16	RET	Return Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{RET} field. 1 Event owned by software debug. DBCR0 _{RET} is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC5 control and status fields. 1 Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC6 control and status fields. 1 Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC7 control and status fields. 1 Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC8 control and status fields. 1 Event owned by software debug. IAC8 control are software readable/writable.

Table 1441. EDBRAC0 field descriptions (continued)

Bit	Name	Description
21	DEVT1	External Debug Event Input 1 Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{DEVT1} field. 1 Event owned by software debug. DBCR0 _{DEVT1} is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{DEVT2} field. 1 Event owned by software debug. DBCR0 _{DEVT2} is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event 0 Event owned by hardware debug. No mtpmr access by software to the PMRs. Performance monitor interrupts set EDBSR0 _{PMI} regardless of the setting of PMGC0 _{UDI} . 1 Event owned by software debug. PMRs are software readable/writable. <i>Note: This bit is reset to '1'.</i>
24	MPU	Memory Protection Unit Debug Event 0 Event owned by hardware debug. No mpuwe access by software to region descriptors that have the DEBUG control bit set to '1' or to the MPU0CSR0 _{DRDEN,DWDEN,IDEN} control bits, unless the CPU is in a debug session (jd_debug_b is asserted). MPU debug events set EDBSR0 _{MPU} , and if not masked by EDBSRMSK0 _{MPU} , one of EDBSR0 _{IAC} , EDBSR0 _{DACR} , or EDBSR0 _{DACW} . MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (jd_debug_b is asserted). 1 Event owned by software debug. All region descriptors and the MPU0CSR0 _{DRDEN,DWDEN,IDEN} control bits are software readable/writable. <i>Note: This bit is reset to '1'.</i>
25	CIRPT	Critical Interrupt Taken Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{CIRPT} field. 1 Event owned by software debug. DBCR0 _{CIRPT} is software readable/writable.
26	CRET	Critical Return Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{CRET} field. 1 Event owned by software debug. DBCR0 _{CRET} is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 DNI resource owned by hardware debug. Execution of an e_dni or se_dni instruction results in entry into debug mode. 1 DNI resource owned by software debug. Execution of an e_dni or se_dni instruction results in either a debug interrupt (DBCR0 _{IDM} =1 and MSR _{DE} =1) or a nop (DBCR0 _{IDM} =0 or MSR _{DE} =0). <i>Note: DNI events are not blocked during a debug session</i>
28	DQM	Data Acquisition Messaging Registers 0 DEVENT _{DQTAG} and DDAM register are exclusively owned by hardware debug. No mtspr access by software to DEVENT _{DQTAG} field or DDAM register. Attempted access by software is ignored. 1 DEVENT _{DQTAG} and DDAM register are owned by software. Software has read/write access to DEVENT _{DQTAG} field and DDAM register.
29:31	—	Reserved

Table 1442 shows which resources are controlled by EDBRAC0 settings.

Table 1442. EDBRAC0 resource control

	EDBCR0 _{EDM}	EDBCR0 _{IDM}	EDBCR0 _{RST}	EDBCR0 _{UDE}	EDBCR0 _{ICMP}	EDBCR0 _{BRT}	EDBCR0 _{IRPT}	EDBCR0 _{TRAP}	EDBCR0 _{IAC1}	EDBCR0 _{IAC2}	EDBCR0 _{IAC3}	EDBCR0 _{IAC4}	EDBCR0 _{IAC5}	EDBCR0 _{IAC6}	EDBCR0 _{IAC7}	EDBCR0 _{IAC8}	EDBCR0 _{DAC1}	EDBCR0 _{DAC34}	EDBCR0 _{DAC2}	EDBCR0 _{RET}	EDBCR0 _{DEVT1}	EDBCR0 _{DEVT2}	EDBCR0 _{PMI}	EDBCR0 _{MPU}	EDBCR0 _{CIRT}	EDBCR0 _{CRET}	EDBCR0 _{DNI}	EDBCR0 _{DQM}	Software Accessible via mtspr, affected by p_reset_b		
0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—		All debug registers	
1	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{IDM}	
1	1	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{RST}	
1	1	1	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{UDE}	
1	1	1	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{ICMP}	
1	1	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{BRT}	
1	1	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{IRPT}	
1	1	1	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{TRAP}	
1	1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC1, DBCR0 _{IAC1} , DBCR1 _{IAC1US·IAC1ER} , DBCR6 _{IAC1XM}	
1	1	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC2, DBCR0 _{IAC2} , DBCR1 _{IAC2US·IAC2ER} , DBCR6 _{IAC2XM}
1	1	1	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR1 _{IAC12M}
1	1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC3, DBCR0 _{IAC3} , DBCR1 _{IAC3US·IAC3ER} , DBCR6 _{IAC3XM}
1	1	1	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC4, DBCR0 _{IAC4} , DBCR1 _{IAC4US·IAC4ER} , DBCR6 _{IAC4XM}
1	1	1	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR1 _{IAC34M}
1	1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC5, DBCR0 _{IAC5} , DBCR5 _{IAC5US·IAC5ER} , DBCR6 _{IAC5XM}
1	1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC6, DBCR0 _{IAC6} , DBCR5 _{IAC6US·IAC6ER} , DBCR6 _{IAC6XM}
1	1	1	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR5 _{IAC56M}
1	1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC7, DBCR0 _{IAC7} , DBCR5 _{IAC7US·IAC7ER} , DBCR6 _{IAC7XM}

Table 1442. EDBRAC0 resource control (continued)

EDBCR0 _{EDM}	EDBRAC0 _{IDM}	EDBRAC0 _{RST}	EDBRAC0 _{UDE}	EDBRAC0 _{ICMP}	EDBRAC0 _{BRT}	EDBRAC0 _{RPT}	EDBRAC0 _{TRAP}	EDBRAC0 _{IAC1}	EDBRAC0 _{IAC2}	EDBRAC0 _{IAC3}	EDBRAC0 _{IAC4}	EDBRAC0 _{IAC5}	EDBRAC0 _{IAC6}	EDBRAC0 _{IAC7}	EDBRAC0 _{IAC8}	EDBRAC0 _{DAC1}	EDBRAC0 _{DAC34}	EDBRAC0 _{DAC2}	EDBRAC0 _{RET}	EDBRAC0 _{DEVT1}	EDBRAC0 _{DEVT2}	EDBRAC0 _{PMI}	EDBRAC0 _{MPU}	EDBRAC0 _{CIRT}	EDBRAC0 _{CRET}	EDBRAC0 _{DNI}	EDBRAC0 _{DQM}	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	IAC8, DBCR0 _{IAC8} , DBCR5 _{IAC8US·IAC8ER} , DBCR6 _{IAC8XM}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	DBCR5 _{IAC78M}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	DAC1, DVC1, DVC1U DBCR0 _{DAC1} , DBCR2 _{DAC1US·DAC1ER} , DBCR2 _{DVC1M·DVC1BE} DBCR4 _{DVC1C,DAC1XM}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DAC3, DAC4 DBCR7 _{DAC{3,4}} , DAC{3,4}CFG, DAC{3,4}XM, DAC{3,4}XMH DBCR8 _{DAC{3,4}US} , DAC{3,4}ER, DAC{3,4}M
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	DAC2, DVC2, DVC2U DBCR0 _{DAC2} , DBCR2 _{DAC2US·DAC2ER} , DBCR2 _{DVC2M·DVC2BE} DBCR4 _{DVC2C,DAC2XM}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR2 _{DAC12M}
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	DBCR2 _{DAC1LNK}
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	DBCR2 _{DAC2LNK}
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	DBCR8 _{DAC3LNK}
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	DBCR8 _{DAC4LNK}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	DBCR0 _{RET}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	DBCR0 _{DEVT1}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	DBCR0 _{DEVT2}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	All PMRs
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	MPU entries with DEBUG bit set, MPU0CSR0 _{DRDEN} , DWDEN, IDEN
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	DBCR0 _{CIRPT}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	DBCR0 _{CRET}
1	(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1 DEVENT _{DQTAG} , DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}=1$. Only software-owned bits or fields as shown in [Table 1442](#) are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

63.3.4 Debug Event Select register (DEVENT) register

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the “Data Acquisition ID Tag Field” section in the Core (e200z420n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in [Figure 1543](#).

Figure 1543. Debug Event Select register (DEVENT) register

DQTAG	0	DEVNT
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31		
SPR 975; Reset ⁽¹⁾ : 0x0		

- Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}=0$, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}=1$, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

[Table 1443](#) provides bit definitions for the Debug Event Register.

Table 1443. DEVENT field descriptions

Bit	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 No signal is asserted xxxxxxx1 p_devnt_out[0] and jd_watchpt[12] are asserted for one clock xxxxxx1x p_devnt_out[1] and jd_watchpt[13] are asserted for one clock xxxxx1xx p_devnt_out[2] and jd_watchpt[20] are asserted for one clock xxxx1xxx p_devnt_out[3] and jd_watchpt[21] are asserted for one clock xxx1xxxx p_devnt_out[4] is asserted for one clock xx1xxxxx p_devnt_out[5] is asserted for one clock x1xxxxxx p_devnt_out[6] is asserted for one clock 1xxxxxxx p_devnt_out[7] is asserted for one clock

63.3.5 Debug Data Acquisition Message (DDAM) register

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the “Data Acquisition Messaging” section in the Core (e200z420n3) Nexus 3 Module chapter for details.

The DDAM register is shown in [Figure 1544](#).

Figure 1544. Debug Data Acquisition Message (DDAM) register

DDAM																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SPR 576; Reset ⁽¹⁾ : 0x0																															

- Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1444](#) provides bit definitions for the Debug Data Acquisition Message Register.

Table 1444. DDAM field descriptions

Bit	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

63.4 Using debug resources for stack limit checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to

perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access that generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in [Table 1435](#). Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the DBCR4_{DAC1CFG} field to '1000' or the DBCR8_{DAC3CFG} field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the DBCR4_{DAC1CFG} field is set to '1000' with DBCR2_{DAC1US} set to '00' or '10', or the DBCR7_{DAC3CFG} field is set to '1000' with DBCR8_{DAC3US} set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the DBCR4_{DAC1CFG} field is set to '1000' with DBCR2_{DAC1US} set to '00' or '01', or the DBCR7_{DAC3CFG} field is set to '1000' with DBCR8_{DAC3US} set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the EDBSR0_{ID} status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow, and for user mode accesses,

a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in [Table 1435](#) and [Table 1438](#) in [Section 63.3.2, Debug Control and Status registers](#). Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

63.5 External debug support

External debug support is supplied through the OnCE controller serial interface, which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the EDBCR0_{EDM}/DBCR0_{EDM} bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When EDBCR0_{EDM} is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of [Section 63.2, Software debug events and exceptions](#) refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note: On the initial setting of EDBCR0_{EDM} to '1', other bits in DBCR0 will remain unchanged. After EDBCR0_{EDM} has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information that could cause unintended operation.

Note: It is intended for the CPU to remain in external debug mode (EDBCR0_{EDM}=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note: *EDBCR0_{EDM} operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if EDBCR0_{EDM} was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all hardware debug events are blocked. Watchpoints are not blocked.*

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the EDBCR0_{EDM} bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

63.5.1 External debug registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug mode.

63.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software. However, the state of EDBCR0_{EDM} is reflected as a read-only bit in DBCR0_{EDM} to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in [Figure 1545](#).

Figure 1545. External Debug Control Register 0 (EDBCR0)

EDM	DNH_EN	0					DNI_CTL	0																							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Reset⁽¹⁾: 0x0

- EDBCR0 is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**.

Table 1445 provides bit definitions for External Debug Control Register 0.

Table 1445. EDBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b . This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0.
1	DNH_EN	dnh Instruction Enable 0 Execution of e_dnh and se_dnh instructions cause illegal instruction exceptions to occur. 1 Execution of e_dnh and se_dnh instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	—	Reserved
7	DNI_CTL	dni Instruction Control 0 When the dni resource is owned by hardware, the MSR _{DE} bit is cared, and when MSR _{DE} =0, execution of e_dni and se_dni instructions are nop'ed and no entry into debug mode occurs. 1 When the dni resource is owned by hardware, the MSR _{DE} bit is don't-cared, and execution of e_dni and se_dni instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR _{DE} . <i>Note: This control bit is only used when the dni resource is owned by hardware via control in EDBRAC0_{DNI}, and thus also only when EDM=1.</i>
8:31	—	Reserved

63.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in Figure 1546.

Figure 1546. External Debug Status Register 0 (EDBSR0)

IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST	0							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Read/Write; Reset⁽¹⁾ - 0x0000 0000

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logics_Reset state or while EDBCR0_{EDM}=0.

Table 1446 provides bit definitions for External Debug Status Register 0.

Table 1446. EDBSR0 field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if EDBCR0 _{EDM} =1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store that is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events that are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with EDBRAC0 _{PMI} =0 regardless of the setting of PMGC0 _{UDI}

Table 1446. EDBSR0 field descriptions (continued)

Bit	Name	Description
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if EDBSRMSK0 _{MPU} is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If EDBSRMSK0 _{MPU} is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by EDBCR0 _{DNH_EN} . Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and EDBSR0 _{IDE} is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

63.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events that are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in [Figure 1547](#).

Figure 1547. External Debug Status Register Mask 0 (EDBSRMSK0)

0	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	0									
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Read/Write; Reset⁽¹⁾ - 0x0000 0000

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while EDBCR0_{EDM}=0.

[Table 1447](#) provides bit definitions for External Debug Status Register Mask 0.

Table 1447. EDBSRMSK0 field descriptions

Bit	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{UDE}
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by EDBSR0 _{DNH}
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{ICMP}
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{BRT}
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{IRPT}
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{TRAP}
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{IAC}
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DACR}
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DACW}
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DNI}
16	RET	Return Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{RET}
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DEVT1}
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DEVT2}
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{PMI}
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{MPU}

Table 1447. EDBSRMSK0 field descriptions (continued)

Bit	Name	Description
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{CIRPT}
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{CRET}
22:31	—	Reserved

63.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of EDBSR0_{DAC{R,W}} bits that are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the EDBSR0_{DAC{R,W}} bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in [Figure 1548](#).

Figure 1548. External Debug Data Effective Address Register (EDDEAR)

Data Effective Address																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Read/Write; Reset: unaffected																															

63.5.2 OnCE introduction

The e200z420n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z420n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated

product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

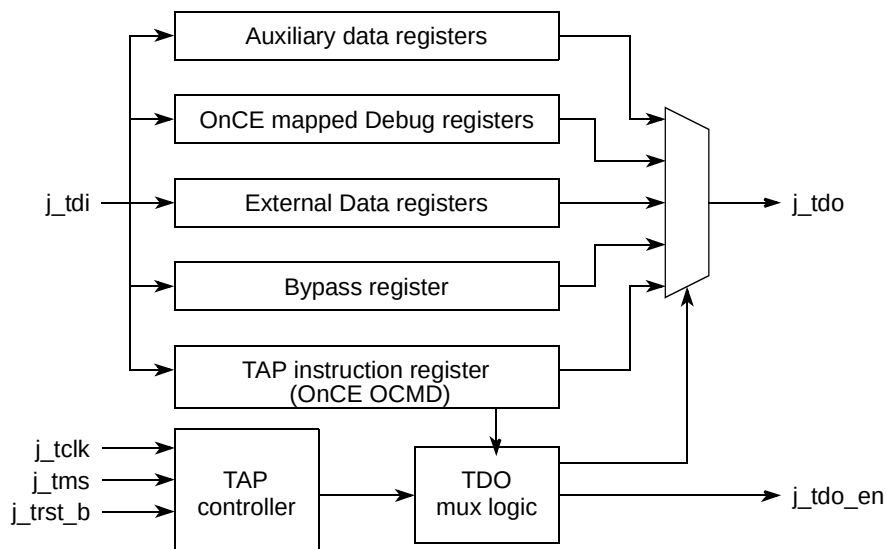


Figure 1549. OnCE TAP controller and registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.

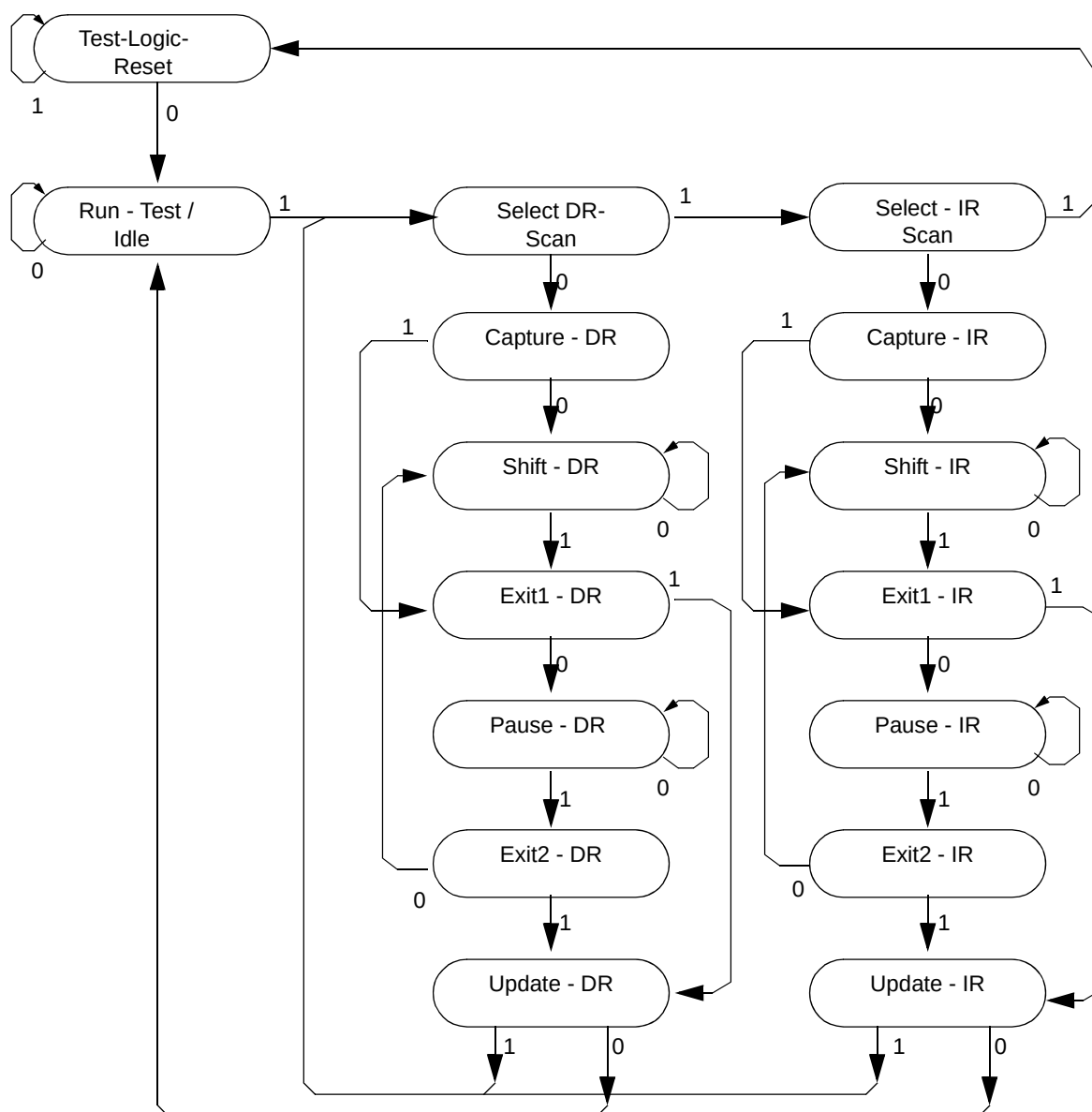


Figure 1550. OnCE controller implementation

Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core, which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), [Section 63.5.6.1, OnCE](#)

[Status register](#)) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will indicate the CPU is in the debug state. Instructions may be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See [Section 63.5.6.2, OnCE Command \(OCMD\) register](#)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

63.5.3 JTAG/OnCE pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

[Table 1448](#) details the primary JTAG/OnCE interface signals.

Table 1448. JTAG/OnCE primary interface signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en ⁽¹⁾	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z420n3) Core Complex Overview.

63.5.4 OnCE internal interface signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

63.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

63.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

63.5.4.3 CPU address and attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

63.5.4.4 CPU data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

63.5.5 OnCE interface signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks that may need information pertaining to debug operation.

63.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regssel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

63.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z420n3 interface) provides a fast means of entering the Debug mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug mode. Once Debug mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems that do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

63.5.5.3 OnCE Debug output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control that are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either “No Register Selected” or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See [Section 63.5.6.2, OnCE Command \(OCMD\) register](#) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

63.5.5.4 CPU Clock On input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z420n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

63.5.5.5 Watchpoint events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to [Table 1455](#) for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

63.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates “no register selected”. This signal will assert regardless of whether the CPU is currently in debug mode. It may be

monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with “no register selected”, and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs that are not in debug mode will ignore the command.

63.5.6 OnCE controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. [Figure 1551](#) is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR, and Update-DR states.

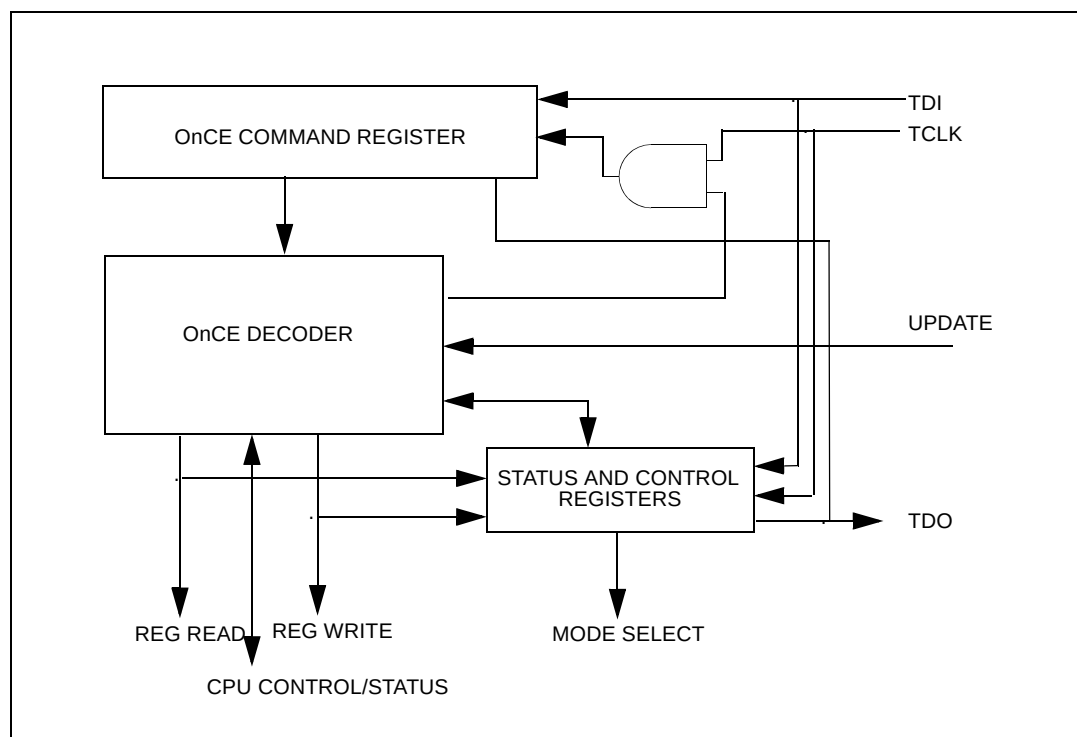


Figure 1551. OnCE Controller and Serial Interface

63.5.6.1 OnCE Status register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

Figure 1552. OnCE Status register

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

[Table 1449](#) provides bit definitions for the OnCE Status register.

Table 1449. OnCE Status register field descriptions

Bit	Name	Description
0	MCLK	MCLK m_clk Status Bit 0 Inactive state 1 Active state This status bit reflects the logic level on the jd_mclk_on input signal after capture by j_tclk .
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
2	—	Reserved, set to zero
3	RESET	RESET Mode This bit reflects the <u>inverted</u> logic level on the CPU p_reset_b input after capture by j_tclk .
4	HALT	HALT Mode This bit reflects the logic level on the CPU p_halted output after capture by j_tclk .
5	STOP	STOP Mode This bit reflects the logic level on the CPU p_stopped output after capture by j_tclk .
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU p_waiting output after capture by j_tclk .
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

63.5.6.2 OnCE Command (OCMD) register

The OnCE Command (OCMD) register is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in [Figure 1553](#). The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

Figure 1553. OnCE Command (OCMD) register

R/W	GO	EX	RS[0:6]						
0	1	2	3	4	5	6	7	8	9

Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por**, or while in the Test Logic Reset state

[Table 1450](#) provides bit definitions for the OCMD.

Table 1450. OCMD field descriptions

Bit	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. <i>Note: The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on j_tdo during the first 32 clocks of Shift-DR.</i> 0 Write the data associated with the command into the register specified by RS[0:6] 1 Read the data contained in the register specified by RS[0:6]
1	GO	Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction that resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. <i>Note: Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Section 63.5.9.6, Exiting Debug mode and interrupt blocking.</i> 0 Inactive (no action taken) 1 Execute instruction in IR

Table 1450. OCMD field descriptions (continued)

Bit	Name	Description
2	EX	Exit Command Bit 0 Remain in debug mode 1 Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. <i>Note: If the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and EDBCR0_{EDM}=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode without execution of an instruction, even though the EX bit was set.</i> <i>Note: Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Section 63.5.9.6, Exiting Debug mode and interrupt blocking.</i>
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 1451 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

[Table 1451](#) indicates the OnCE register addresses.

Table 1451. OnCE register addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)

Table 1451. OnCE register addressing (continued)

RS[0:6]	Register Selected
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) — *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) — *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select

Table 1451. OnCE register addressing (continued)

RS[0:6]	Register Selected
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3 Access (see Chapter Chapter 14: e200z4201n3 Nexus 3 Module)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE ⁽¹⁾
111 1111	Bypass

1. Causes assertion of the `j_en_once_regsel` output.

The OnCE Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

Single-stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The `OSR_ERR` bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the `EDBCR0_EDM` bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of `EDBCR0_EDM`. DBSR may still be modified during a debug session via a single-stepped `mtspr` instruction, or via OnCE access.

If the CPU is not currently in debug mode, `go+exit` and `go+noexit` commands are ignored, although for a `go+exit` command with "No Register Selected", the `j_ocmd_go_exit` output will be asserted.

63.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z420n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see [Section 63.7, MPU operation during debug](#)). The control bits are read/write. These bits are only effective while OnCE is enabled (`jd_en_once` asserted). The OCR is shown in [Figure 1554](#).

Figure 1554. OnCE Control Register (OCR)

0								I_DMDIS	0	I_DVLE	I_DI	0	D_DMDIS	0	D_DI	0	D_DG	0									WKUP	FDB	DR		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Reset - 0xo000_0000 on **m_por**, **j_trst_b**, or entering Test logic Reset state

Table 1452 provides bit definitions for the OnCE Control Register.

Table 1452. OCR field descriptions

Bit	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to '1' to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug 'VLE' Attribute Bit This bit is used to provide the 'VLE' attribute bit to be used during a debug session. <i>Note: This bit is ignored, since VLE is always enabled.</i>
12	I_DI	Instruction Side Debug 'I' Attribute Bit This bit is used to provide the 'I' attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to '1' to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
17:19	—	Reserved
20	D_DI	Data Side Debug 'I' Attribute Bit This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
21	—	Reserved

Table 1452. OCR field descriptions (continued)

Bit	Name	Description
22	D_DG	Data Side Debug 'G' Attribute Bit This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit This control bit may be used to force the p_wakeup output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the m_clk input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the ' bkpt ' pseudo-instruction will cause the processor to enter debug mode, as if the jd_de_b input had been asserted. This bit is qualified with EDBCR0_EDM , which must be set for FDB to take effect. <i>Note: This bit has no effect on e_dnh or se_dnh instruction operation.</i>
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter Debug mode. The CPU will indicate that Debug mode has been entered via the data scanned out in the shift-IR state. 0 No Debug mode request 1 Unconditional Debug mode request When the DR bit is set, the processor will enter Debug mode at the next instruction boundary.

63.5.7 Access to debug resources

Resources contained in the OnCE Module that do not require the processor core to be halted for access may be accessed while the e200z420n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z420n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z420n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the **OCR_WKUP** control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

[Table 1453](#) provides a list of access requirements for OnCE registers.

Table 1453. OnCE register access requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for write access	Requires CPU to be halted for read access	Requires CPU to be halted for write access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*(1)	
DBCR0	Y	Y	Y	N	*(1)	*EDBCR0 _{EDM} access only requires jd_en_once asserted
DBCR1–8	Y	Y	Y	N	*(1)	
DEVENT	Y	Y	Y	N	*(1)	
EDBRAC0 (DBERC0)	Y	N	Y	N	*(1)	
DBSR	Y	Y	Y	N ⁽²⁾	*(1)	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*(1)	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while jd_en_once is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

63.5.8 Methods of entering Debug mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See [Section 63.5.5.1, OnCE Enable \(jd_en_once\)](#).

63.5.8.1 External debug request during RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b**, and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z420n3 core to enter Debug mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z420n3 core does not execute an instruction before entering Debug mode, although the first instruction to be executed will be fetched prior to entering Debug mode in order to properly initialize the CPUSCR.

63.5.8.2 Debug request during RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

63.5.8.3 Debug request during normal activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

63.5.8.4 Debug request during Waiting, Halted, or Stopped state

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting**, **p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** output during the Shift_IR TAP controller state. This status may be

used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

63.5.8.5 Software request during normal activity

Upon executing a **'bkpt'** pseudo-instruction (for e200z420n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and EDBCR0_{EDM}=1, the CPU enters the debug mode after the instruction following the **'bkpt'** pseudo-instruction has entered the instruction register.

63.5.8.6 Debug notify halt instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by EDBCR0_{DNH_EN}, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is **not** conditioned on EDBCR0_{EDM}, allowing for debug of software debug handlers as well as other software. For e200z420n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

63.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. [Figure 1555](#) shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

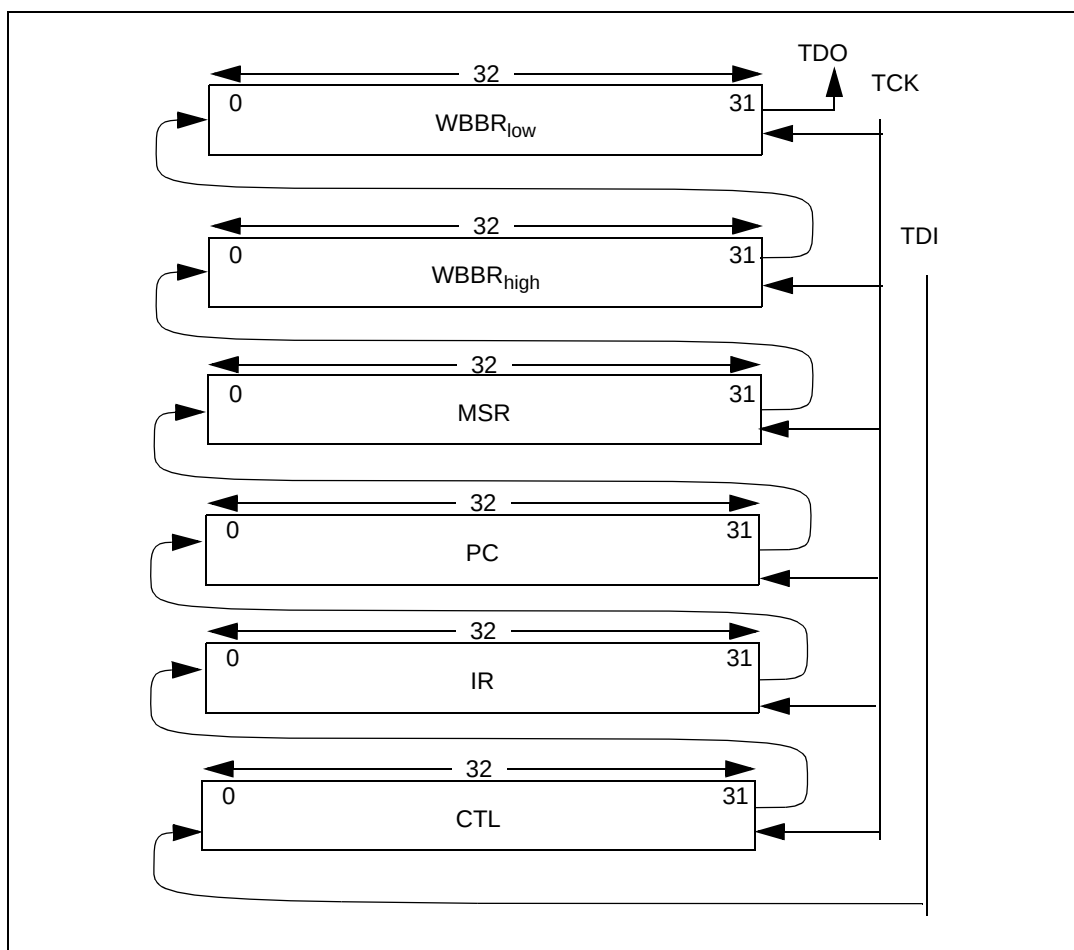


Figure 1555. CPU Scan Chain Register (CPUSCR)

63.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

63.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the

operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

Figure 1556. Control State Register (CTL)

*										IRSTAT14	IRSTAT13	IRSTAT12	IRSTAT11	IRSTAT10	WAITING	PCOFST				PCINV	FFRA	IRSTAT0	IRSTAT1	IRSTAT2	IRSTAT3	IRSTAT4	IRSTAT5	IRSTAT6	IRSTAT7	IRSTAT8	IRSTAT9
										10	11	12	13	14	15					16	17	18	19	20	21	22	23	24	25	26	27

Table 1454. CTL field descriptions

Bit	Name	Description
0:9	*	Internal State Bits These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See Section 63.5.5.3, OnCE Debug output (jd_debug_b) , which is not part of the normal program execution flow, these bits should be set to a 0.
10	IRSTAT14	IR Status Bit 14 This control bit indicates an MPU Instruction Address Compare event status for the IR. 0 No MPU Instruction Address Compare event occurred on the fetch of this instruction. 1 An MPU Instruction Address Compare event occurred on the fetch of this instruction.
11	IRSTAT13	IR Status Bit 13 This control bit indicates an Instruction Address Compare 8 event status for the IR. 0 No Instruction Address Compare 8 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 8 event occurred on the fetch of this instruction.
12	IRSTAT12	IR Status Bit 12 This control bit indicates an Instruction Address Compare 7 event status for the IR. 0 No Instruction Address Compare 7 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 7 event occurred on the fetch of this instruction.
13	IRSTAT11	IR Status Bit 11 This control bit indicates an Instruction Address Compare 6 event status for the IR. 0 No Instruction Address Compare 6 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 6 event occurred on the fetch of this instruction.
14	IRSTAT10	IR Status Bit 10 This control bit indicates an Instruction Address Compare 5 event status for the IR. 0 No Instruction Address Compare 5 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 5 event occurred on the fetch of this instruction.

Table 1454. CTL field descriptions (continued)

Bit	Name	Description
15	WAITING	WAITING State Status This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit. 0 CPU was not in the waiting state when debug mode was entered 1 CPU was in the waiting state when debug mode was entered
16:19	PCOFST	PC Offset Field This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV, which overrides this field). 0000 No correction required. 0001 Subtract 0x04 from PC. 0010 Subtract 0x08 from PC. 0011 Subtract 0x0C from PC. 0100 Subtract 0x10 from PC. 0101 Subtract 0x14 from PC. All other encodings are reserved
20	PCINV	PC and IR Invalid Status Bit This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered. 0 No error condition exists. 1 Error condition exists. PC and IR are corrupted.
21	FFRA	Feed Forward RA Operand Bit This control bit causes the content of the WBBR_{lo} to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR. For most LSP instructions using rA rB as a 64-bit source operand, WBBR_{hi}, lo is used to supply the 64-bit source value. This allows the debug firmware to update processor registers — initialize the WBBR_{lo} with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register. Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent. 0 No action. 1 Content of WBBR_{lo} used as operand value.

Table 1454. CTL field descriptions (continued)

Bit	Name	Description
22	IRSTAT0	IR Status Bit 0 This control bit indicates a TEA status for the IR. 0 No TEA occurred on the fetch of this instruction. 1 TEA occurred on the fetch of this instruction.
23	IRSTAT1	IR Status Bit 1 This control bit is reserved.
24	IRSTAT2	IR Status Bit 2 This control bit indicates an Instruction Address Compare 1 event status for the IR. 0 No Instruction Address Compare event 1 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 1 occurred on the fetch of this instruction.
25	IRSTAT3	IR Status Bit 3 This control bit indicates an Instruction Address Compare 2 event status for the IR. 0 No Instruction Address Compare 2 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 2 event occurred on the fetch of this instruction.
26	IRSTAT4	IR Status Bit 4 This control bit indicates an Instruction Address Compare 3 event status for the IR. 0 No Instruction Address Compare event 3 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 3 occurred on the fetch of this instruction.
27	IRSTAT5	IR Status Bit 5 This control bit indicates an Instruction Address Compare 4 event status for the IR. 0 No Instruction Address Compare 4 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 4 event occurred on the fetch of this instruction.
28	IRSTAT6	IR Status Bit 6 This control bit indicates a Parity Error status for the IR. 0 No Parity Error occurred on the fetch of this instruction. 1 Parity Error occurred on the fetch of this instruction from the I-Cache or the IMEM.
29	IRSTAT7	IR Status Bit 7 This control bit indicates a Precise External Termination Error status for the IR. 0 No Precise External Termination Error occurred on the fetch of this instruction. 1 A Precise External Termination Error occurred on the fetch of this instruction.
30	IRSTAT8	IR Status Bit 8 This control bit indicates the PowerISA VLE status for the IR. 0 IR contains a BookE instruction. (unused encoding) 1 IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit. <i>Note: This bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).</i>
31	IRSTAT9	IR Status Bit 9 This control bit is reserved.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the

desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit that was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC + 4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events that may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be reentered prior to any further instruction execution.

63.5.9.3 Program Counter (PC) register

The PC is a 32-bit register that stores the value of the program counter that was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

63.5.9.4 Write-Back Bus Register (WBBR_{low}, WBBR_{high})

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. WBBR_{low} holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. WBBR_{high} holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into WBBR_{low}. The contents of WBBR_{low} can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed, which uses this value as a substitute data value. The Control State

register FFRA bit forces the value of the WBBR_{low} to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to [Section 63.5.9.2, Control State Register \(CTL\)](#) for more detail on the CTL_{FFRA} bit.)

WBBR_{low} and WBBR_{high} are generally undefined on instructions that do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

63.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used. This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

63.5.9.6 Exiting Debug mode and interrupt blocking

When exiting debug mode with a Go+Exit, “asynchronous” interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will reenter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

63.5.10 Reserved registers (Reserved)

The reserved registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

63.6 Watchpoint support

e200z420n3 supports the generation and signalling of watchpoints when operating in internal debug mode (DBCR0_{IDM}=1) or in external debug mode (EDBCR0_{EDM}=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with EDBCR0_{EDM} or DBCR0_{IDM}.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVENT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31]**) for various development control purposes. (See the “Watchpoint Trace Messaging” section in the Core (e200z420n3) Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

[Table 1455](#) shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 1455. Watchpoint output signal assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on DBCR4 _{DAC1CFG} settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on DBCR4 _{DAC2CFG} settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on DBCR7 _{DAC3CFG} settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on DBCR7 _{DAC4CFG} settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status

Table 1455. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever p_pm_event transitions from a '0' to a '1' while m_clk is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on PMLCa0 _{PMP}
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on PMLCa1 _{PMP}
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on PMLCa2 _{PMP}

Table 1455. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on PMLCa3 _{PMP}
jd_watchpt[27]	MPU	Memory Protection Unit watchpoint Asserted whenever a the MPU generates a debug event based on region descriptor matches with DEBUG control enabled.
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

63.7 MPU operation during debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR ([Section 63.5.6.3, OnCE Control Register \(OCR\)](#) for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR ([Section 63.5.6.3, OnCE Control Register \(OCR\)](#) for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

63.8 Cache array access during debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

63.9 Basic steps for enabling, using, and exiting External Debug mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug mode and initializing debug registers:

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR} , OCR_{WKUP} are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set.

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in [Section 63.5.9.2, Control State Register \(CTL\)](#)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with “no-register selected”, Go cleared, and determines that the PCU has reentered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in [Section 63.5.9.2, Control State Register \(CTL\)](#)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the EDBCR0_{EDM} bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should rewrite the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in [Section 63.5.9.2, Control State Register \(CTL\)](#)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note: These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

64 e200z425n3 Core Debug Support

This chapter describes the debug features of the e200z425n3 core.

64.1 Overview

Internal debug support in the e200z425n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware-based debugger, which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *PowerISA 2.06*. e200z425n3 supports a subset of these defined facilities. In addition to the facilities defined in *PowerISA 2.06*, e200z425n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

64.1.1 Software debug facilities

e200z425n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors that are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 (DBCR0_{IDM}). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When DBCR0_{IDM} is cleared (and EDBCR0_{EDM} is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when DBCR0_{IDM} is cleared (or is overridden by EDBCR0_{EDM} being set and EDBRAC0 indicating no resource is “owned” by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

64.1.1.1 *PowerISA 2.06* compatibility

The e200z425n3 core implements a subset of the *PowerISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

64.1.2 Additional debug facilities

In addition to the debug functionality defined in *PowerISA 2.06*, e200z425n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See [Section 64.1.4, *Sharing debug resources by software/hardware*](#).) A data trace port is also provided.

64.1.2.1 Data trace port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z425n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the “Data Trace Control Register (DTC)” section in the Core (e200z425n3) Nexus 3 Module chapter. The data trace port will report aligned accesses, and most misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0. Also, for accesses generated by a LSP load/store instruction using circular addressing mode that crosses a buffer boundary, the two portions of the misaligned access will be reported separately, since otherwise it is not possible to correctly correlate the data with the initial access address.

64.1.3 Hardware debug facilities

The e200z425n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z425n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug mode enable bit in External Debug Control register 0 (EDBCR0_{EDM}), which is also aliased to DBCR0_{EDM}. Setting EDBCR0_{EDM} overrides the Internal Debug mode enable bit DBCR0_{IDM} unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the “hardware-owned” debug facilities. In addition, since resources are “owned” by the hardware debugger, inconsistent values may be present if software attempts to read “hardware-owned” debug-related resources.

When hardware debug is enabled by setting EDBCR0_{EDM}=1, the control registers and resources described in [Section 64.3, *Debug registers*](#) are reserved for use by the external debugger. The same events described in [Section 64.2, *Software debug events and*](#)

exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. EDBCRO_{EDM}, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

64.1.4 Sharing debug resources by software/hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When EDBCRO_{EDM} is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources that set DBSR bits when DBCRO_{IDM}=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources that set EDBSR0 bits when EDBCRO_{EDM}=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, (EDBCRO_{EDM}=1 and EDBRAC0_{IDM}=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

64.1.4.1 Simultaneous hardware and software debug event handling

Since it is possible that a “hardware-owned” resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented that guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRRO/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

Figure 1557 shows the e200z425n3 debug resources.

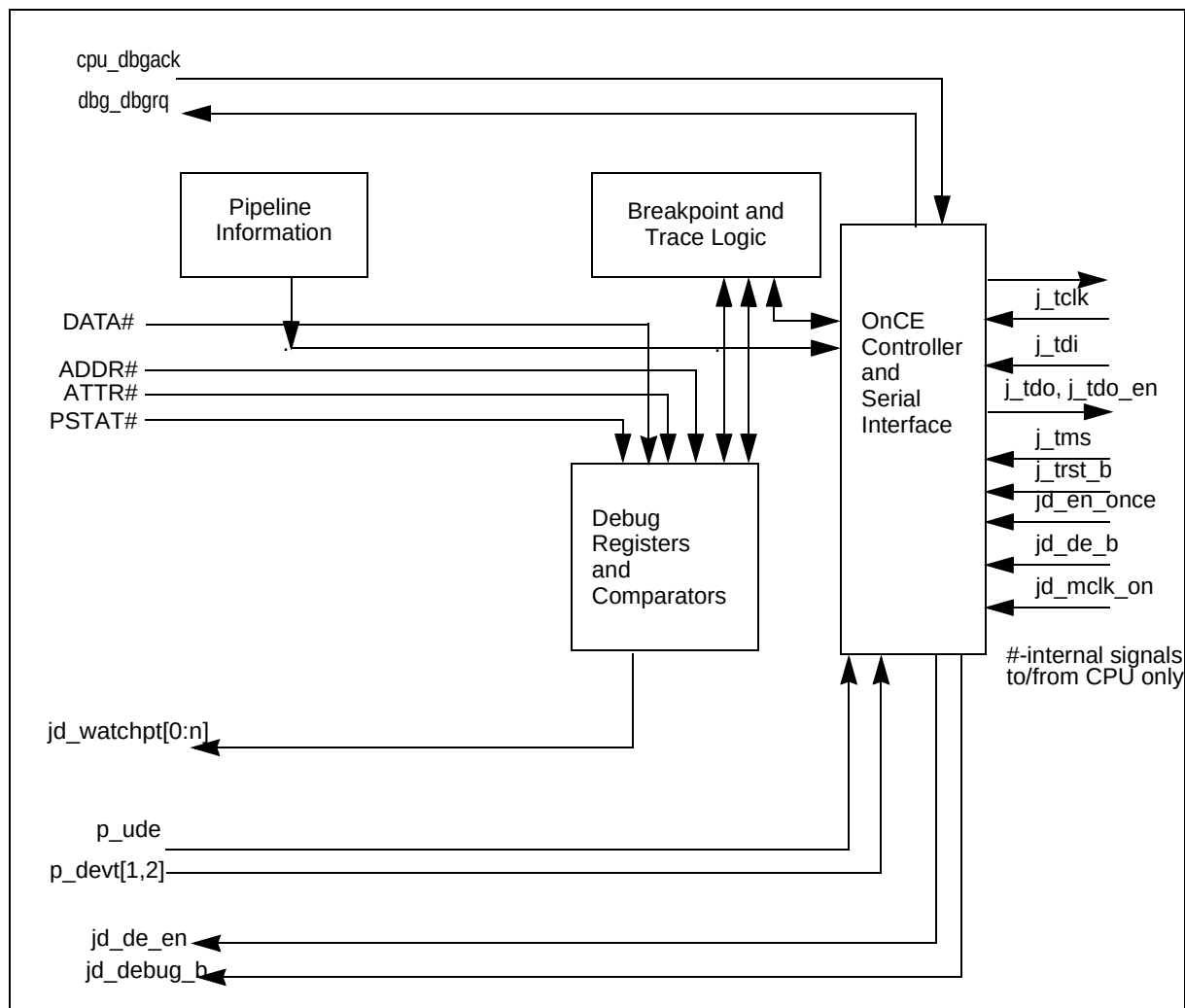


Figure 1557. e200z425n3 debug resources

64.2 Software debug events and exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}=1$) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in $EDBRAC0$). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers ($DBCR0-8$). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource that is “owned” by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE} , a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when $MSR_{DE}=0$ and $DBCR0_{IDM}=1$. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE} . A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while $MSR_{DE}=0$, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction that enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** that causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *PowerISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z425n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2, which are described in [Section 64.2.11, External debug event](#).
- The Performance Monitor Interrupt event PMI, which is described in [Section 64.2.13, Performance Monitor Interrupt debug event](#).

The e200z425n3 debug configuration supports most of these event types. Unsupported *PowerISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

64.2.1 Instruction Address Compare event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

64.2.2 Data Address Compare event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . Four address compare values (DAC1–4) are provided.

The effective address of the load or store operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note: In contrast to the PowerISA 2.06 definition, Data Address Compare events on e200z425n3 do not prevent the load or store class instruction from completing. If a load or store class instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}=1$, this circumstance can also occur when $EDBCR0_{EDM}=1$ and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note: .DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note: DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.

Note: DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note: Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

64.2.2.1 Data Address Compare event status updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class

instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC_OFST}$ field when the DVC DAC status is recorded.

[Table 1456](#) outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in [Table 1456](#), the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 1456. DAC events and resultant updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. <i>Note: in this case the 2nd Ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>

Table 1456. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
DVC DACx	DVC DACy, Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
DVC DACx	DVC DACy, Normal Ldst	Ldst instruction, no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
DVC DACx	DVC DACy, Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i> <i>Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
DVC DACx	DVC DACy, Normal Ldst	DACy, or DVC DACy, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
DVC DACx	DVC DACy, Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1456. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	Any instruction (no exception)	DSI, with or without DAC, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. <i>Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
DVC DACx	Any instruction (no exception)	DACy, or DVC DACy Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1457 through Table 1460 show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 1457. DAC events and resultant updates, dual-issue case 1

Instruction sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)

Table 1457. DAC events and resultant updates, dual-issue case 1 (continued)

Instruction sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.</i> <i>Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.</i>

Table 1458. DAC events and resultant updates, dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1458. DAC events and resultant updates, dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i> <i>Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). <i>Note: In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1459. DAC events and resultant updates, dual-issue case 3

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.

Table 1459. DAC events and resultant updates, dual-issue case 3 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) <i>Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

Table 1460. DAC events and resultant updates, dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.

Table 1460. DAC events and resultant updates, dual-issue case 4 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). <i>Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.</i>
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). <i>Note: in this case if $x=y$, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). <i>Note: in this case if $x=y$, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.</i>

64.2.3 Linked Instruction Address Compare and Data Address Compare events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note: *Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.*

64.2.4 Trap debug event

A Trap debug event (TRAP) occurs if Trap debug events are enabled (DBCR0_{TRAP}=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the DBSR_{TRAP} bit is set to 1 to record the debug exception.

64.2.5 Branch Taken debug event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled (DBCR0_{BRT}=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus DBSR_{ID}E can not be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the DBSR_{BRT} bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

64.2.6 Instruction Complete debug event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled (DBCR0_{ICMP}=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The **se_sc** instruction does not fall into the category of an instruction whose execution is suppressed, since the

instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in $DSRR0$.

Instruction Complete debug events are not recognized if $MSR_{DE}=0$ at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed ($SRR0$ points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with $SRR0$ pointing to the instruction following the instruction that generated the EFPU Round exception, and $DSRR0$ will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}=1$, this circumstance can also occur when $EDBCR0_{EDM}=1$ and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}=1$, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while $MSR_{DE}=1$.

64.2.7 Interrupt Taken debug event

An Interrupt Taken debug event ($IRPT$) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}=1$) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using $SRR0/1$) cause an Interrupt Taken debug event. This event can occur and be recorded in $DBSR$ regardless of the setting of MSR_{DE} . When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in $DSRR0$ will be the address of the non-critical interrupt handler.

64.2.8 Critical Interrupt Taken debug event

A Critical Interrupt Taken debug event ($CIRPT$) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}=1$) and a critical interrupt occurs. Only critical class interrupts (an interrupt using $CSRR0/1$) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in $DBSR$ regardless of the setting of MSR_{DE} . When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in $DSRR0$ will be the address of the critical interrupt handler.

64.2.9 Return debug event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}=1$) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in $DBSR$ regardless of the setting of MSR_{DE} . When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If $MSR_{DE}=0$ at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If $MSR_{DE}=1$ at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

64.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}=1$) and an attempt is made to execute an **se_rfc**i instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If $MSR_{DE}=0$ at the time of the execution of the **se_rfc**i (i.e. before the MSR is updated by the **se_rfc**i), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If $MSR_{DE}=1$ at the time of the execution of the **se_rfc**i, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfc**i instruction.

64.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}=1$ or $DBCR0_{DEVT2}=1$), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

64.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

64.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}=1$), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE} . When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

64.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register $EDBRAC0$, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in $EDBRAC0$ corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause

modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not “owned” by software is not guaranteed to be consistent if accessed (read) by software with a **mfspir** instruction, other than the DBCR0_{EDM} bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

64.3.1 Debug Address and Value registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in [Figure 1558](#)) corresponds to the byte data value transferred to/from memory byte offset 0, 8, ..., and the least significant byte of the register (labeled B7 in [Figure 1558](#)) corresponds to byte offset 7, F, When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison; they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z425n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspir** and **mfspir** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

Figure 1558. DVC1, DVC2 registers

B0								B1								B2								B3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
B4								B5								B6								B7							
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

SPR 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset: Unaffected

64.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z425n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

64.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z425n3 adds some implementation specific bits to this register, as seen in [Figure 1559](#).

Figure 1559. Debug Control Register 0 (DBCR0) register

EDM	IDM	RST		ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC2		RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	0		CIRPT	CRET	0					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 308; Read/Write; Reset⁽¹⁾ - 0x0

1. DBCR0_{EDM} is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if DBCR0_{EDM}=0, as well as unconditionally by **m_por**. If DBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the DBCR0_{RST} field will be reset by **p_reset_b**. The DBCR0_{RST} field will always be reset by **p_reset_b** regardless of the value of DBCR0_{EDM}.

[Table 1461](#) provides bit definitions for Debug Control Register 0.

Table 1461. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRO–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If $MSR_{DE}=1$, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 p_dbrstc[1] pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 p_dbrstc[0] pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled

Table 1461. DBCR0 field descriptions (continued)

Bit	Name	Description
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1 IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled

Table 1461. DBCR0 field descriptions (continued)

Bit	Name	Description
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

64.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in [Figure 1560](#).

Figure 1560. DBCR1 register

IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	0	IAC3US	IAC3ER	IAC4US	IAC4ER	IAC34M	0
--------	--------	--------	--------	--------	---	--------	--------	--------	--------	--------	---

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 309; Read/Write; Reset⁽¹⁾ - 0x0

- Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1462](#) provides bit definitions for Debug Control Register 1.

Table 1462. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR _{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR _{PR} =1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC1 debug events are based on effective address and can only occur if MSR _{IS} =1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR _{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR _{PR} =1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC2 debug events are based on effective address and can only occur if MSR _{IS} =1

Table 1462. DBCR1 field descriptions (continued)

Bit	Name	Description
8:9	IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

64.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation. The DBCR2 register is shown in [Figure 1561](#).

Figure 1561. DBCR2 register

DAC1US	DAC1ER	DAC2US	DAC2ER	DAC12M	DAC1LNK	DAC2LNK	DVC1M	DVC2M	DVC1BE					DVC2BE																	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 310: Read/Write: Reset⁽¹⁾ - 0x0

- Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1463](#) provides bit definitions for Debug Control Register 2.

Table 1463. DBCR2 field descriptions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR _{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR _{PR} =1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR _{DS} =0 11 DAC1 debug events are based on effective address and can only occur if MSR _{DS} =1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR _{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR _{PR} =1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR _{DS} =0 11 DAC2 debug events are based on effective address and can only occur if MSR _{DS} =1

Table 1463. DBCR2 field descriptions (continued)

Bit	Name	Description
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies <i>Exact Address Compare</i> since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR.

Table 1463. DBCR2 field descriptions (continued)

Bit	Name	Description
12:13	DVC1M	Data Value Compare 1 Mode When DBCR4_{DVC1C}=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. <i>Note: Inactive byte lanes of the memory access are automatically masked.</i> When DBCR4_{DVC1C}=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved <i>Note: Inactive byte lanes of the memory access are automatically masked.</i>
14:15	DVC2M	Data Value Compare 2 Mode When DBCR4_{DVC2C}=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. <i>Note: Inactive byte lanes of the memory access are automatically masked.</i> When DBCR4_{DVC2C}=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved <i>Note: Inactive byte lanes of the memory access are automatically masked.</i>

Table 1463. DBCR2 field descriptions (continued)

Bit	Name	Description
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

64.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in [Figure 1562](#).

Figure 1562. DBCR4 register

0				0								0	DAC1XMH	0	DAC2XMH	DAC1XM				DAC2XM				DAC1CFG				DAC2CFG			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 563; Read/Write; Reset⁽¹⁾ - 0x0

1. DBCR4 is reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1464](#) provides bit definitions for Debug Control Register 4.

Table 1464. DBCR4 field descriptions

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.

Table 1464. DBCR4 field descriptions (continued)

Bit	Name	Description
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}=00$ 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}=00$ 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}=00$ 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). <i>Note: Unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}=0$; when $DBCR0_{DAC1} \neq 0$, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.</i>
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}=00$ 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}=00$ 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}=00$ 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. <i>Note: Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}=00$. When $DBCR0_{DAC2} \neq 0$, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.</i>

64.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in [Figure 1563](#)

Figure 1563. DBCR5 register

IAC5US	IAC5ER	IAC6US	IAC6ER	IAC56M	0	IAC7US	IAC7ER	IAC8US	IAC8ER	IAC78M	0
--------	--------	--------	--------	--------	---	--------	--------	--------	--------	--------	---

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 564; Read/Write; Reset⁽¹⁾ - 0x0

1. Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1462 provides bit definitions for Debug Control Register 5.

Table 1465. DBCR5 field descriptions

Bit	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR _{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR _{PR} =1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC5 debug events are based on effective address and can only occur if MSR _{IS} =1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR _{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR _{PR} =1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC6 debug events are based on effective address and can only occur if MSR _{IS} =1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved

Table 1465. DBCR5 field descriptions (continued)

Bit	Name	Description
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR _{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR _{PR} =1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC7 debug events are based on effective address and can only occur if MSR _{IS} =1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR _{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR _{PR} =0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR _{PR} =1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR _{IS} =0 11 IAC8 debug events are based on effective address and can only occur if MSR _{IS} =1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

64.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in [Figure 1564](#).

Figure 1564. DBCR6 register

IAC1XM				IAC2XM				IAC3XM				IAC4XM				IAC5XM				IAC6XM				IAC7XM				IAC8XM			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 603; Read/Write; Reset⁽¹⁾ - 0x0

1. DBCR6 is reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1466 provides bit definitions for Debug Control Register 6.

Table 1466. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 = Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Control. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

Table 1466. DBCR6 field descriptions (continued)

Bit	Name	Description
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101 – 1111 = Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

64.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in [Figure 1565](#).

Figure 1565. DBCR7 register

0	0	DAC3XMH	0	DAC4XMH	DAC3XM	DAC4XM	DAC3CFG	DAC4CFG
---	---	---------	---	---------	--------	--------	---------	---------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 596; Read/Write; Reset⁽¹⁾ - 0x0

1. DBCR7 is reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1467](#) provides bit definitions for Debug Control Register 7.

Table 1467. DBCR7 field descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High. DAC3XMH extends the range of the DAC3XM field. 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits

Table 1467. DBCR7 field descriptions (continued)

Bit	Name	Description
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001–1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in [Figure 1566](#).

DAC3US	DAC3ER	DAC4US	DAC4ER	DAC34M	DAC3LNK	DAC4LNK	0																								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 597; Read/Write; Reset⁽¹⁾ - 0x0

1. Reset by processor reset **p_reset_b** if EDBCR0_EDM=0, as well as unconditionally by **m_por**. If EDBCR0_EDM=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1468. DBCR8 field descriptions

Bit	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 DAC3 debug events not affected by MSR _{PR} 01 Reserved 10 DAC3 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 DAC3 debug events can only occur if MSR _{PR} =1 (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 DAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC3 debug events are based on effective address and can only occur if MSR _{DS} =0 11 DAC3 debug events are based on effective address and can only occur if MSR _{DS} =1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 DAC4 debug events not affected by MSR _{PR} 01 Reserved 10 DAC4 debug events can only occur if MSR _{PR} =0 (Supervisor mode) 11 DAC4 debug events can only occur if MSR _{PR} =1 (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 DAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC4 debug events are based on effective address and can only occur if MSR _{DS} =0 11 DAC4 debug events are based on effective address and can only occur if MSR _{DS} =1

Table 1468. DBCR8 field descriptions (continued)

Bit	Name	Description
8:9	DAC34M	Data Address Compare 3/4 Mode 00 Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 No effect 1 DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 No effect 1 DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

64.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr DBSR,RS**. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug mode is enabled ($DBCR0_{IDM}=1$). When debug interrupts are enabled ($MSR_{DE}=1$ $DBCR0_{IDM}=1$ and $EDBCR0_{EDM}=0$, or $MSR_{DE}=1$, $DBCR0_{IDM}=1$, $EDBCR0_{EDM}=1$ and software is allocated resource(s) via $EDBRAC0$), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}=1$ and $EDBRAC0_{IDM}=1$), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in [Figure 1567](#).

Figure 1567. Debug Status Register (DBSR)

IDE	UDE	MRR	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST	0								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 304; Read/Clear; Reset - 0x1000 0000

Table 1469 provides bit definitions for the Debug Status Register.

Table 1469. DBSR field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $MSR_{DE}=0$ and $DBCR0_{IDM}=1$ and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store that is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 No reset occurred since these bits were last cleared by software 01 A hard reset occurred since these bits were last cleared by software 10 Reserved 11 Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred

Table 1469. DBSR field descriptions (continued)

Bit	Name	Description
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UD}[1]=1$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

64.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load or store operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in [Figure 1568](#).

Figure 1568. Debug Data Effective Address Register (DDEAR)

Data Effective Address																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 600; Read/Write; Reset - unaffected

64.3.3 External Debug Resource Allocation Control (EDBRAC0) register

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}=1$. EDBRAC0 settings are ignored when $EDBCR0_{EDM}=0$.

Hardware-owned resources that generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources that generate debug events if $DBCR0_{IDM}=1$ update DBSR, causing debug interrupts to occur if $MSR_{DE}=1$. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug mode is enabled. When debug interrupts are enabled ($MSR_{DE}=1$, $DBCR0_{IDM}=1$ and $EDBCR0_{EDM}=0$, or $MSR_{DE}=1$, $DBCR0_{IDM}=1$ and $EDBCR0_{EDM}=1$ and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event that is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug mode is enabled ($EDBCR0_{EDM}=1$). When $EDBCR0_{EDM}=1$, a set bit in EDBSR0 by an event that is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}=1$, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in $DBCR0-8$ or $MPU0CSR0$ are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in [Figure 1569](#).

Figure 1569. External Debug Resource Allocation Control (EDBRAC0) register

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	DNI	DQM	0
---	-----	-----	-----	------	-----	------	------	------	------	------	------	------	-------	------	---	-----	------	------	------	------	-------	-------	-----	-----	-------	------	-----	-----	---

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b**, reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

[Table 1469](#) provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 1470. EDBRAC0 field descriptions

Bit	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control. When EDBRAC0 _{IDM} =1, software writes to hardware-owned bits in DBCR0–8 via mtspr are ignored. 0 Internal Debug mode may not be enabled by software. DBCR0 _{IDM} is owned exclusively by hardware. mtspr DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 Internal Debug mode may be enabled by software. DBCR0 _{IDM} is owned by software. DBCR0 _{IDM} is software readable/writable.
2	RST	Reset Field Control 0 DBCR0 _{RST} owned exclusively by hardware debug. No mtspr access by software to DBCR0 _{RST} field. 1 DBCR0 _{RST} accessible by software debug. DBCR0 _{RST} is software readable/writable.
3	UDE	Unconditional Debug Event 0 Event owned by hardware debug. 1 Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{ICMP} field. 1 Event owned by software debug. DBCR0 _{ICMP} is software readable/writable.
5	BRT	Branch Taken Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{BRT} field. 1 Event owned by software debug. DBCR0 _{BRT} is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{IRPT} field. 1 Event owned by software debug. DBCR0 _{IRPT} is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{TRAP} field. 1 Event owned by software debug. DBCR0 _{TRAP} is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC1 control and status fields. 1 Event owned by software debug. IAC1 control fields are software readable/writable.

Table 1470. EDBRAC0 field descriptions (continued)

Bit	Name	Description
9	IAC2	Instruction Address Compare 2 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC2 control and status fields. 1 Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC3 control and status fields. 1 Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC4 control and status fields. 1 Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 Event owned by hardware debug. No mtspr access by software to DAC1 control and status fields. 1 Event owned by software debug. DAC1 control fields are software readable/writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 Events owned by hardware debug. No mtspr access by software to DAC3 and DAC4 control and status fields. 1 Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 Event owned by hardware debug. No mtspr access by software to DAC2 control and status fields. 1 Event owned by software debug. DAC2 control fields are software readable/writable.
15	—	Reserved
16	RET	Return Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{RET} field. 1 Event owned by software debug. DBCR0 _{RET} is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC5 control and status fields. 1 Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC6 control and status fields. 1 Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC7 control and status fields. 1 Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 Event owned by hardware debug. No mtspr access by software to IAC8 control and status fields. 1 Event owned by software debug. IAC8 control are software readable/writable.

Table 1470. EDBRAC0 field descriptions (continued)

Bit	Name	Description
21	DEVT1	External Debug Event Input 1 Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{DEVT1} field. 1 Event owned by software debug. DBCR0 _{DEVT1} is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{DEVT2} field. 1 Event owned by software debug. DBCR0 _{DEVT2} is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event 0 Event owned by hardware debug. No mtpmr access by software to the PMRs. Performance monitor interrupts set EDBSR0 _{PMI} regardless of the setting of PMGC0 _{UDI} . 1 Event owned by software debug. PMRs are software readable/writable. <i>Note: This bit is reset to '1'.</i>
24	MPU	Memory Protection Unit Debug Event 0 Event owned by hardware debug. No mpuwe access by software to region descriptors that have the DEBUG control bit set to '1' or to the MPU0CSR0 _{DRDEN,DWDEN,IDEN} control bits, unless the CPU is in a debug session (jd_debug_b is asserted). MPU debug events set EDBSR0 _{MPU} , and if not masked by EDBSRMSK0 _{MPU} , one of EDBSR0 _{IAC} , EDBSR0 _{DACR} , or EDBSR0 _{DACW} . MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (jd_debug_b is asserted). 1 Event owned by software debug. All region descriptors and the MPU0CSR0 _{DRDEN,DWDEN,IDEN} control bits are software readable/writable. <i>Note: This bit is reset to '1'.</i>
25	CIRPT	Critical Interrupt Taken Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{CIRPT} field. 1 Event owned by software debug. DBCR0 _{CIRPT} is software readable/writable.
26	CRET	Critical Return Debug Event 0 Event owned by hardware debug. No mtspr access by software to DBCR0 _{CRET} field. 1 Event owned by software debug. DBCR0 _{CRET} is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 DNI resource owned by hardware debug. Execution of an e_dni or se_dni instruction results in entry into debug mode. 1 DNI resource owned by software debug. Execution of an e_dni or se_dni instruction results in either a debug interrupt (DBCR0 _{IDM} =1 and MSR _{DE} =1) or a nop (DBCR0 _{IDM} =0 or MSR _{DE} =0). <i>Note: DNI events are not blocked during a debug session</i>
28	DQM	Data Acquisition Messaging Registers 0 DEVENT _{DQTAG} and DDAM register are exclusively owned by hardware debug. No mtspr access by software to DEVENT _{DQTAG} field or DDAM register. Attempted access by software is ignored. 1 DEVENT _{DQTAG} and DDAM register are owned by software. Software has read/write access to DEVENT _{DQTAG} field and DDAM register.
29:31	—	Reserved

Table 1471 shows which resources are controlled by EDBRAC0 settings.

Table 1471. EDBRAC0 resource control

	EDBCR0 _{EDM}	EDBRAC0 _{IDM}	EDBRAC0 _{RST}	EDBRAC0 _{UDE}	EDBRAC0 _{ICMP}	EDBRAC0 _{BRT}	EDBRAC0 _{IRPT}	EDBRAC0 _{TRAP}	EDBRAC0 _{IAC1}	EDBRAC0 _{IAC2}	EDBRAC0 _{IAC3}	EDBRAC0 _{IAC4}	EDBRAC0 _{IAC5}	EDBRAC0 _{IAC6}	EDBRAC0 _{IAC7}	EDBRAC0 _{IAC8}	EDBRAC0 _{DAC1}	EDBRAC0 _{DAC34}	EDBRAC0 _{DAC2}	EDBRAC0 _{RET}	EDBRAC0 _{DEVT1}	EDBRAC0 _{DEVT2}	EDBRAC0 _{PMI}	EDBRAC0 _{MPU}	EDBRAC0 _{GIRT}	EDBRAC0 _{CRET}	EDBRAC0 _{DNI}	EDBRAC0 _{DOM}	Software Accessible via mtspr, affected by p_reset_b
0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	All debug registers
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{IDM}
1	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{RST}
1	1	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{UDE}
1	1	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{ICMP}
1	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{BRT}
1	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{IRPT}
1	1	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR0 _{TRAP}
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC1, DBCR0 _{IAC1} , DBCR1 _{IAC1US·IAC1ER} , DBCR6 _{IAC1XM}
1	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC2, DBCR0 _{IAC2} , DBCR1 _{IAC2US·IAC2ER} , DBCR6 _{IAC2XM}
1	1	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR1 _{IAC12M}
1	1	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC3, DBCR0 _{IAC3} , DBCR1 _{IAC3US·IAC3ER} , DBCR6 _{IAC3XM}
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC4, DBCR0 _{IAC4} , DBCR1 _{IAC4US·IAC4ER} , DBCR6 _{IAC4XM}
1	1	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR1 _{IAC34M}

Table 1471. EDBRAC0 resource control (continued)

EDBCR0 _{EDM}	EDBRAC0 _{DM}	EDBRAC0 _{RST}	EDBRAC0 _{UDE}	EDBRAC0 _{ICMP}	EDBRAC0 _{BRT}	EDBRAC0 _{IRPT}	EDBRAC0 _{TRAP}	EDBRAC0 _{IAC1}	EDBRAC0 _{IAC2}	EDBRAC0 _{IAC3}	EDBRAC0 _{IAC4}	EDBRAC0 _{IAC5}	EDBRAC0 _{IAC6}	EDBRAC0 _{IAC7}	EDBRAC0 _{IAC8}	EDBRAC0 _{DAC1}	EDBRAC0 _{DAC34}	EDBRAC0 _{DAC2}	EDBRAC0 _{RET}	EDBRAC0 _{DEVT1}	EDBRAC0 _{DEVT2}	EDBRAC0 _{PMI}	EDBRAC0 _{MPU}	EDBRAC0 _{CIRT}	EDBRAC0 _{CRET}	EDBRAC0 _{DNI}	EDBRAC0 _{DOM}	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC5, DBCR0 _{IAC5} , DBCR5 _{IAC5US·IAC5ER} , DBCR6 _{IAC5XM}
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC6, DBCR0 _{IAC6} , DBCR5 _{IAC6US·IAC6ER} , DBCR6 _{IAC6XM}
1	1	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR5 _{IAC56M}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC7, DBCR0 _{IAC7} , DBCR5 _{IAC7US·IAC7ER} , DBCR6 _{IAC7XM}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	IAC8, DBCR0 _{IAC8} , DBCR5 _{IAC8US·IAC8ER} , DBCR6 _{IAC8XM}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	DBCR5 _{IAC78M}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	DAC1, DVC1, DVC1U DBCR0 _{DAC1} , DBCR2 _{DAC1US·DAC1E} R, DBCR2 _{DVC1M·DVC1BE} DBCR4 _{DVC1C,DAC1XM}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DAC3, DAC4 DBCR7 _{DAC{3,4}} , DAC{3,4}CFG, DAC{3,4}XM, DAC{3,4}XMH DBCR8 _{DAC{3,4}US} , DAC{3,4}ER, DAC{3,4}M
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	DAC2, DVC2, DVC2U DBCR0 _{DAC2} , DBCR2 _{DAC2US·DAC2E} R, DBCR2 _{DVC2M·DVC2BE} DBCR4 _{DVC2C,DAC2XM}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	

Table 1471. EDBRAC0 resource control (continued)

	EDBCR0 _{EDM}	EDBRAC0 _{IDM}	EDBRAC0 _{RST}	EDBRAC0 _{UDE}	EDBRAC0 _{ICMP}	EDBRAC0 _{BRT}	EDBRAC0 _{IRPT}	EDBRAC0 _{TRAP}	EDBRAC0 _{JAC1}	EDBRAC0 _{JAC2}	EDBRAC0 _{JAC3}	EDBRAC0 _{JAC4}	EDBRAC0 _{JAC5}	EDBRAC0 _{JAC6}	EDBRAC0 _{JAC7}	EDBRAC0 _{JAC8}	EDBRAC0 _{DAC1}	EDBRAC0 _{DAC34}	EDBRAC0 _{DAC2}	EDBRAC0 _{RET}	EDBRAC0 _{DEVT1}	EDBRAC0 _{DEVT2}	EDBRAC0 _{PMI}	EDBRAC0 _{MPU}	EDBRAC0 _{CIRT}	EDBRAC0 _{CRET}	EDBRAC0 _{DNI}	EDBRAC0 _{DOM}	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	1	—	—	—	—	—	—	—	—	—	DBCR2 _{DAC12M}
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	DBCR2 _{DAC1LNK}
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DBCR2 _{DAC2LNK}
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DBCR8 _{DAC3LNK}
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DBCR8 _{DAC4LNK}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	DBCR0 _{RET}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	DBCR0 _{DEVT1}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	DBCR0 _{DEVT2}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	All PMRs
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	MPU entries with DEBUG bit set, MPU0CSR0 _{DRDEN} , DWDEN, IDEN
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	DBCR0 _{CIRPT}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	DBCR0 _{CRET}
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	
1	(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	DEVENT _{DQTAG} DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when EDBCR0_{EDM}=1. Only software-owned bits or fields as shown in [Table 1471](#) are affected in this case, except that DBCR0_{RST} and DBCR0_{MRR} are updated by assertion of **p_reset_b** regardless of the value of EDBCR0_{EDM} or EDBRAC0.

64.3.4 Debug Event Select register (DEVENT) register

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering.

The DEVENT_{DEVNT} register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z425n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in [Figure 1570](#).

Figure 1570. Debug Event Select register (DEVENT) register

DQTAG	0	DEVNT
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31		

SPR 975; Reset⁽¹⁾: 0x0

- Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

[Table 1472](#) provides bit definitions for the Debug Event Register.

Table 1472. DEVENT field descriptions

Bit	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 No signal is asserted xxxxxx1x p_devnt_out[0] and jd_watchpt[12] are asserted for one clock xxxxxx1x p_devnt_out[1] and jd_watchpt[13] are asserted for one clock xxxxx1xx p_devnt_out[2] and jd_watchpt[20] are asserted for one clock xxxx1xxx p_devnt_out[3] and jd_watchpt[21] are asserted for one clock xxx1xxxx p_devnt_out[4] is asserted for one clock xx1xxxxx p_devnt_out[5] is asserted for one clock x1xxxxxx p_devnt_out[6] is asserted for one clock 1xxxxxxx p_devnt_out[7] is asserted for one clock

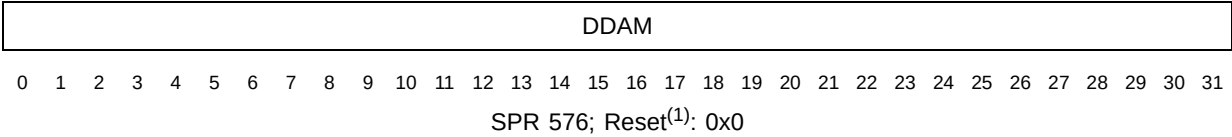
64.3.5 Debug Data Acquisition Message (DDAM) register

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this

register. See the “Data Acquisition Messaging” section in the Core (e200z425n3) Nexus 3 Module chapter for details.

The DDAM register is shown in [Figure 1571](#).

Figure 1571. Debug Data Acquisition Message (DDAM) register



- 1. Reset by processor reset **p_reset_b** if EDBCR0_{EDM}=0, as well as unconditionally by **m_por**. If EDBCR0_{EDM}=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

[Table 1473](#) provides bit definitions for the Debug Data Acquisition Message Register.

Table 1473. DDAM field descriptions

Bit	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

64.4 Using debug resources for stack limit checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0



pointing to the address of the load or store access that generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in [Table 1464](#). Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the DBCR4_{DAC1CFG} field to '1000' or the DBCR8_{DAC3CFG} field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the DBCR4_{DAC1CFG} field is set to '1000' with DBCR2_{DAC1US} set to '00' or '10', or the DBCR7_{DAC3CFG} field is set to '1000' with DBCR8_{DAC3US} set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the DBCR4_{DAC1CFG} field is set to '1000' with DBCR2_{DAC1US} set to '00' or '01', or the DBCR7_{DAC3CFG} field is set to '1000' with DBCR8_{DAC3US} set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the EDBSR0_{IDE} status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by

hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in [Table 1464](#) and [Table 1467](#) in [Section 64.3.2, Debug Control and Status registers](#). Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

64.5 External debug support

External debug support is supplied through the OnCE controller serial interface, which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the EDBCR0_{EDM}/DBCR0_{EDM} bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When EDBCR0_{EDM} is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of [Section 64.2, Software debug events and exceptions](#) refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note: *On the initial setting of EDBCR0_{EDM} to '1', other bits in DBCR0 will remain unchanged. After EDBCR0_{EDM} has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information that could cause unintended operation.*

Note: *It is intended for the CPU to remain in external debug mode (EDBCR0_{EDM}=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.*

Note: *EDBCR0_{EDM} operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if EDBCR0_{EDM} was previously*

set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all hardware debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the EDBCR0_{EDM} bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

64.5.1 External debug registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug mode.

64.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software. However, the state of EDBCR0_{EDM} is reflected as a read-only bit in DBCR0_{EDM} to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in [Figure 1572](#).

Figure 1572. External Debug Control Register 0 (EDBCR0)

EDM		DNH_EN		0				DNI_CTL		0																					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Reset⁽¹⁾: 0x0

- EDBCR0 is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**.

[Table 1474](#) provides bit definitions for External Debug Control Register 0.

Table 1474. EDBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b . This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0.
1	DNH_EN	dnh Instruction Enable 0 Execution of e_dnh and se_dnh instructions cause illegal instruction exceptions to occur. 1 Execution of e_dnh and se_dnh instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	—	Reserved
7	DNI_CTL	dni Instruction Control 0 When the dni resource is owned by hardware, the MSR _{DE} bit is cared, and when MSR _{DE} =0, execution of e_dni and se_dni instructions are nop'ed and no entry into debug mode occurs. 1 When the dni resource is owned by hardware, the MSR _{DE} bit is don't-cared, and execution of e_dni and se_dni instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR _{DE} . <i>Note: This control bit is only used when the dni resource is owned by hardware via control in EDBRAC0_{DNI}, and thus also only when EDM=1.</i>
8:31	—	Reserved

64.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in [Figure 1573](#).

Figure 1573. External Debug Status Register 0 (EDBSR0)

IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST	0							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Read/Write; Reset⁽¹⁾ - 0x0000 0000

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logics_Reset state or while EDBCR0_{EDM}=0.

[Table 1475](#) provides bit definitions for External Debug Status Register 0.

Table 1475. EDBSR0 field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if EDBCR0 _{EDM} =1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store that is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events that are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with EDBRAC0 _{PMI} =0 regardless of the setting of PMGC0 _{UDI}

Table 1475. EDBSR0 field descriptions (continued)

Bit	Name	Description
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if EDBSRMSK0 _{MPU} is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If EDBSRMSK0 _{MPU} is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by EDBCR0 _{DNH_EN} . Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and EDBSR0 _{IDE} is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

64.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events that are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in [Figure 1574](#).

Figure 1574. External Debug Status Register Mask 0 (EDBSRMSK0)

0	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	0									
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Read/Write; Reset⁽¹⁾ - 0x0000 0000

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while EDBCR0_{EDM}=0.

[Table 1476](#) provides bit definitions for External Debug Status Register Mask 0.

Table 1476. EDBSRMSK0 field descriptions

Bit	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{UDE}
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by EDBSR0 _{DNH}
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{ICMP}
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{BRT}
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{IRPT}
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{TRAP}
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{IAC}
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DACR}
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DACW}
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DNI}
16	RET	Return Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{RET}
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DEVT1}
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{DEVT2}
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{PMI}
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{MPU}

Table 1476. EDBSRMSK0 field descriptions (continued)

Bit	Name	Description
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{CIRPT}
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by EDBSR0 _{CRET}
22:31	—	Reserved

64.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load or store operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of EDBSR0_{DAC{R,W}} bits that are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the EDBSR0_{DAC{R,W}} bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in [Figure 1575](#).

Figure 1575. External Debug Data Effective Address Register (EDDEAR)

Data Effective Address																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Read/Write; Reset: unaffected																															

64.5.2 OnCE introduction

The e200z425n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z425n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated

product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

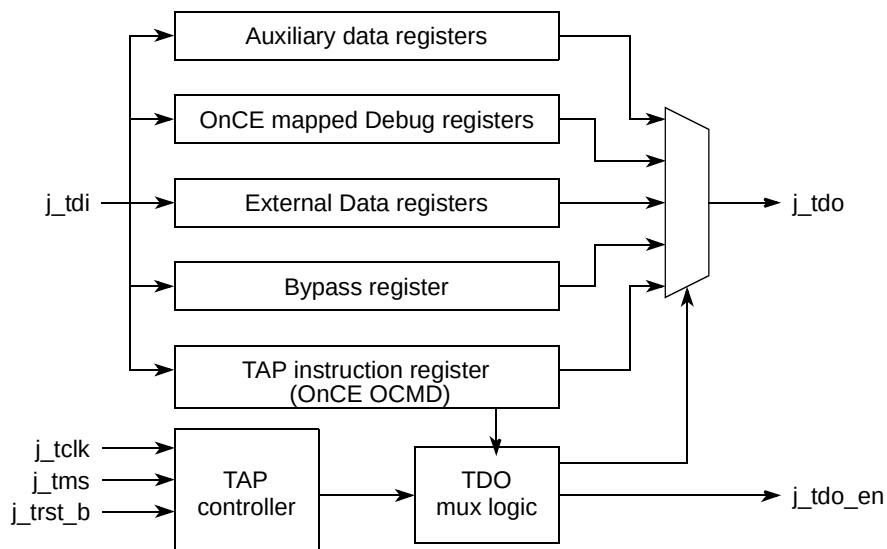


Figure 1576. OnCE TAP controller and registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.

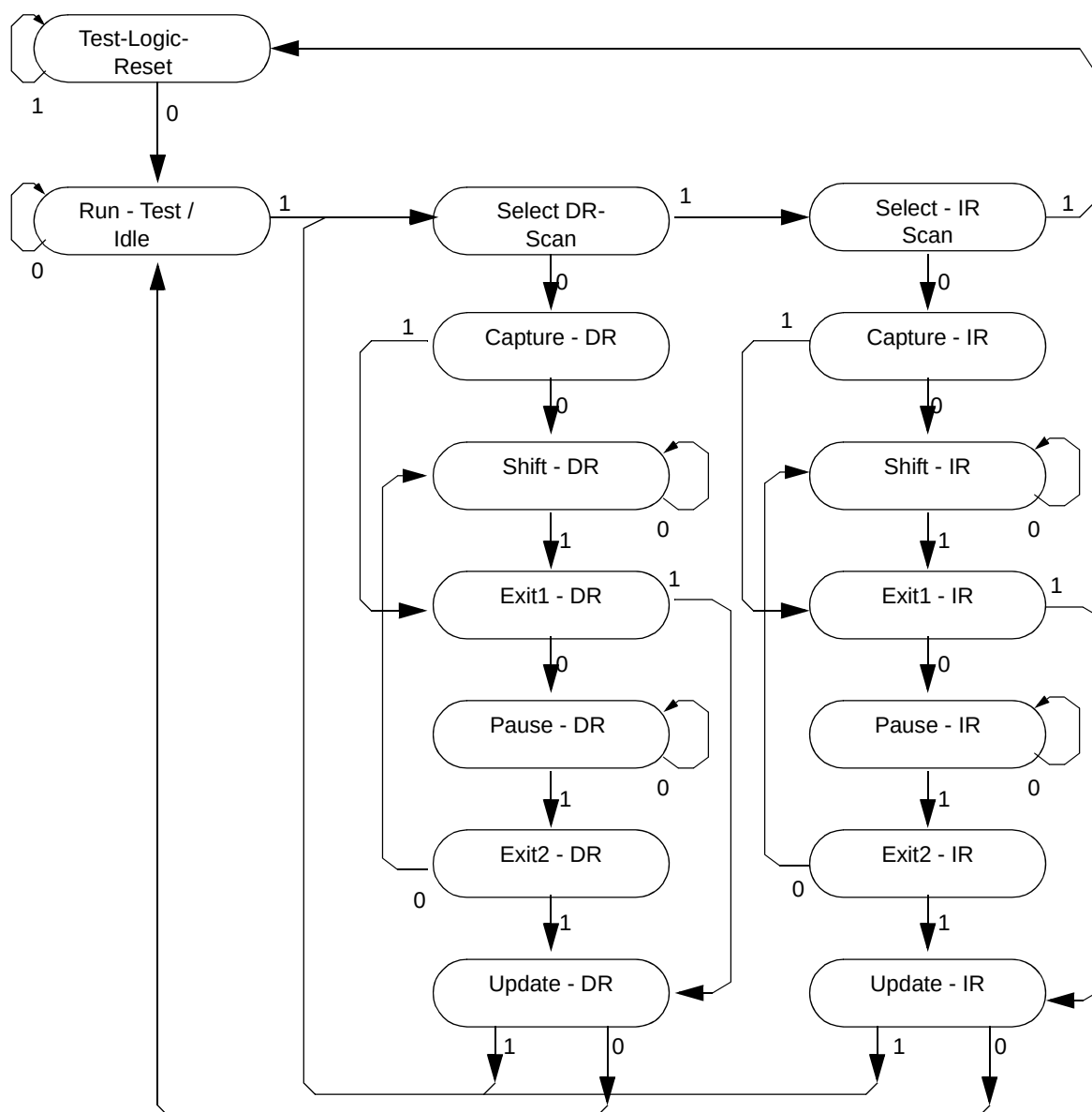


Figure 1577. OnCE controller implementation

Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core, which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), [Section 64.5.6.1, OnCE](#)

Status register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will indicate the CPU is in the debug state. Instructions may be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See [Section 64.5.6.2, OnCE Command \(OCMD\) register](#)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

64.5.3 JTAG/OnCE pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

[Table 1477](#) details the primary JTAG/OnCE interface signals.

Table 1477. JTAG/OnCE primary interface signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en ⁽¹⁾	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z425n3) Core Complex Overview.

64.5.4 OnCE internal interface signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

64.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

64.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

64.5.4.3 CPU address and attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

64.5.4.4 CPU data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

64.5.5 OnCE interface signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks that may need information pertaining to debug operation.

64.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regssel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

64.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z425n3 interface) provides a fast means of entering the Debug mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug mode. Once Debug mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems that do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

64.5.5.3 OnCE Debug output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control that are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either “No Register Selected” or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See [Section 64.5.6.2, OnCE Command \(OCMD\) register](#) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

64.5.5.4 CPU Clock On input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z425n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

64.5.5.5 Watchpoint events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/D TSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to [Table 1484](#) for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

64.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates “no register selected”. This signal will assert regardless of whether the CPU is currently in debug mode. It may be

monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with “no register selected”, and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs that are not in debug mode will ignore the command.

64.5.6 OnCE controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. [Figure 1578](#) is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR, and Update-DR states.

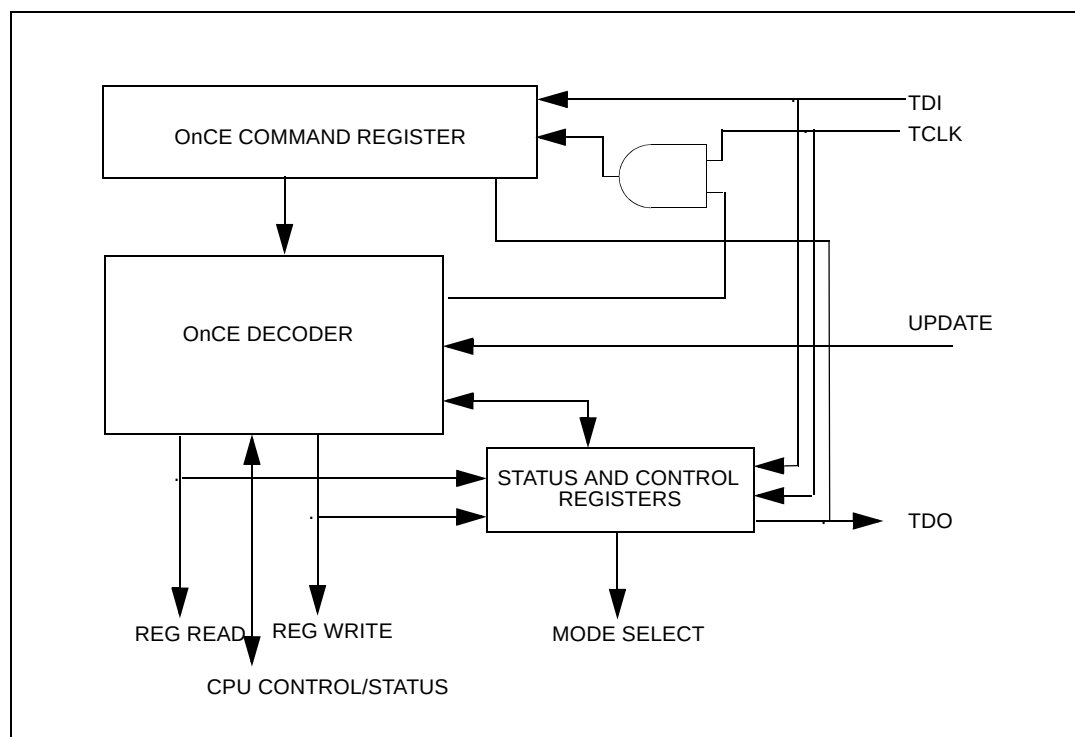


Figure 1578. OnCE Controller and Serial Interface

64.5.6.1 OnCE Status register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

Figure 1579. OnCE Status register

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

[Table 1478](#) provides bit definitions for the OnCE Status register.

Table 1478. OnCE Status register field descriptions

Bit	Name	Description
0	MCLK	MCLK m_clk Status Bit 0 Inactive state 1 Active state This status bit reflects the logic level on the jd_mclk_on input signal after capture by j_tclk.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
2	—	Reserved, set to zero
3	RESET	RESET Mode This bit reflects the <u>inverted</u> logic level on the CPU p_reset_b input after capture by j_tclk.
4	HALT	HALT Mode This bit reflects the logic level on the CPU p_halted output after capture by j_tclk.
5	STOP	STOP Mode This bit reflects the logic level on the CPU p_stopped output after capture by j_tclk.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU p_waiting output after capture by j_tclk.
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

64.5.6.2 OnCE Command (OCMD) register

The OnCE Command (OCMD) register is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in [Figure 1580](#). The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

Figure 1580. OnCE Command (OCMD) register

R/W	GO	EX	RS[0:6]						
0	1	2	3	4	5	6	7	8	9

Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por**, or while in the Test_Logic_Reset state

[Table 1479](#) provides bit definitions for the OCMD.

Table 1479. OCMD field descriptions

Bit	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. <i>Note: The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on j_tdo during the first 32 clocks of Shift-DR.</i> 0 Write the data associated with the command into the register specified by RS[0:6] 1 Read the data contained in the register specified by RS[0:6]
1	GO	Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction that resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. <i>Note: Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Section 64.5.9.6, Exiting Debug mode and interrupt blocking.</i> 0 Inactive (no action taken) 1 Execute instruction in IR

Table 1479. OCMD field descriptions (continued)

Bit	Name	Description
2	EX	Exit Command Bit 0 Remain in debug mode 1 Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. <i>Note: If the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and EDBCR0_{EDM}=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode without execution of an instruction, even though the EX bit was set.</i> <i>Note: Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Section 64.5.9.6, Exiting Debug mode and interrupt blocking.</i>
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 1480 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

[Table 1480](#) indicates the OnCE register addresses.

Table 1480. OnCE register addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)

Table 1480. OnCE register addressing (continued)

RS[0:6]	Register Selected
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) — *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) — *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Reserved for Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select

Table 1480. OnCE register addressing (continued)

RS[0:6]	Register Selected
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3 Access (see Chapter Chapter 14: e200z4201n3 Nexus 3 Module)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE ⁽¹⁾
111 1111	Bypass

1. Causes assertion of the j_en_once_regssel output.

The OnCE Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

Single-stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the EDBCR0_{EDM} bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0_{EDM}. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

64.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z425n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see [Section 64.7, MPU operation during debug](#)). The control bits are read/write. These bits are only effective while OnCE is enabled (j_d_en_once asserted). The OCR is shown in [Figure 1581](#).

Figure 1581. OnCE Control Register (OCR)

0								I_DMDIS	0	I_DVLE	I_DI	0	D_DMDIS	0	D_DI	0	D_DG	0	0								WKUP	FDB	DR		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Reset - 0xo000_0000 on **m_por**, **j_trst_b**, or entering Test logic Reset state

Table 1481 provides bit definitions for the OnCE Control Register.

Table 1481. OCR field descriptions

Bit	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to '1' to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug 'VLE' Attribute Bit This bit is used to provide the 'VLE' attribute bit to be used during a debug session. <i>Note: This bit is ignored, since VLE is always enabled.</i>
12	I_DI	Instruction Side Debug 'I' Attribute Bit This bit is used to provide the 'I' attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to '1' to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
17:19	—	Reserved
20	D_DI	Data Side Debug 'I' Attribute Bit This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
21	—	Reserved

Table 1481. OCR field descriptions (continued)

Bit	Name	Description
22	D_DG	Data Side Debug 'G' Attribute Bit This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit This control bit may be used to force the p_wakeup output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the m_clk input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the ' bkpt ' pseudo-instruction will cause the processor to enter debug mode, as if the jd_de_b input had been asserted. This bit is qualified with EDBCR0_EDM , which must be set for FDB to take effect. <i>Note: This bit has no effect on e_dnh or se_dnh instruction operation.</i>
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter Debug mode. The CPU will indicate that Debug mode has been entered via the data scanned out in the shift-IR state. 0 No Debug mode request 1 Unconditional Debug mode request When the DR bit is set, the processor will enter Debug mode at the next instruction boundary.

64.5.7 Access to debug resources

Resources contained in the OnCE Module that do not require the processor core to be halted for access may be accessed while the e200z425n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z425n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z425n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the **OCR_WKUP** control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

[Table 1482](#) provides a list of access requirements for OnCE registers.

Table 1482. OnCE register access requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for write access	Requires CPU to be halted for read access	Requires CPU to be halted for write access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*(1)	
DBCR0	Y	Y	Y	N	*(1)	*EDBCR0 _{EDM} access only requires jd_en_once asserted
DBCR1–8	Y	Y	Y	N	*(1)	
DEVENT	Y	Y	Y	N	*(1)	
EDBRAC0 (DBERC0)	Y	N	Y	N	*(1)	
DBSR	Y	Y	Y	N ⁽²⁾	*(1)	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*(1)	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while jd_en_once is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

64.5.8 Methods of entering Debug mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See [Section 64.5.5.1, OnCE Enable \(jd_en_once\)](#).

64.5.8.1 External debug request during RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b**, and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z425n3 core to enter Debug mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z425n3 core does not execute an instruction before entering Debug mode, although the first instruction to be executed will be fetched prior to entering Debug mode in order to properly initialize the CPUSCR.

64.5.8.2 Debug request during RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

64.5.8.3 Debug request during normal activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

64.5.8.4 Debug request during Waiting, Halted, or Stopped state

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting**, **p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** output during the Shift_IR TAP controller state. This status may be

used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

64.5.8.5 Software request during normal activity

Upon executing a '**bkpt**' pseudo-instruction (for e200z425n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and EDBCR0_{EDM}=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.

64.5.8.6 Debug notify halt instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by EDBCR0_{DNH_EN}, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is **not** conditioned on EDBCR0_{EDM}, allowing for debug of software debug handlers as well as other software. For e200z425n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

64.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. [Figure 1582](#) shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

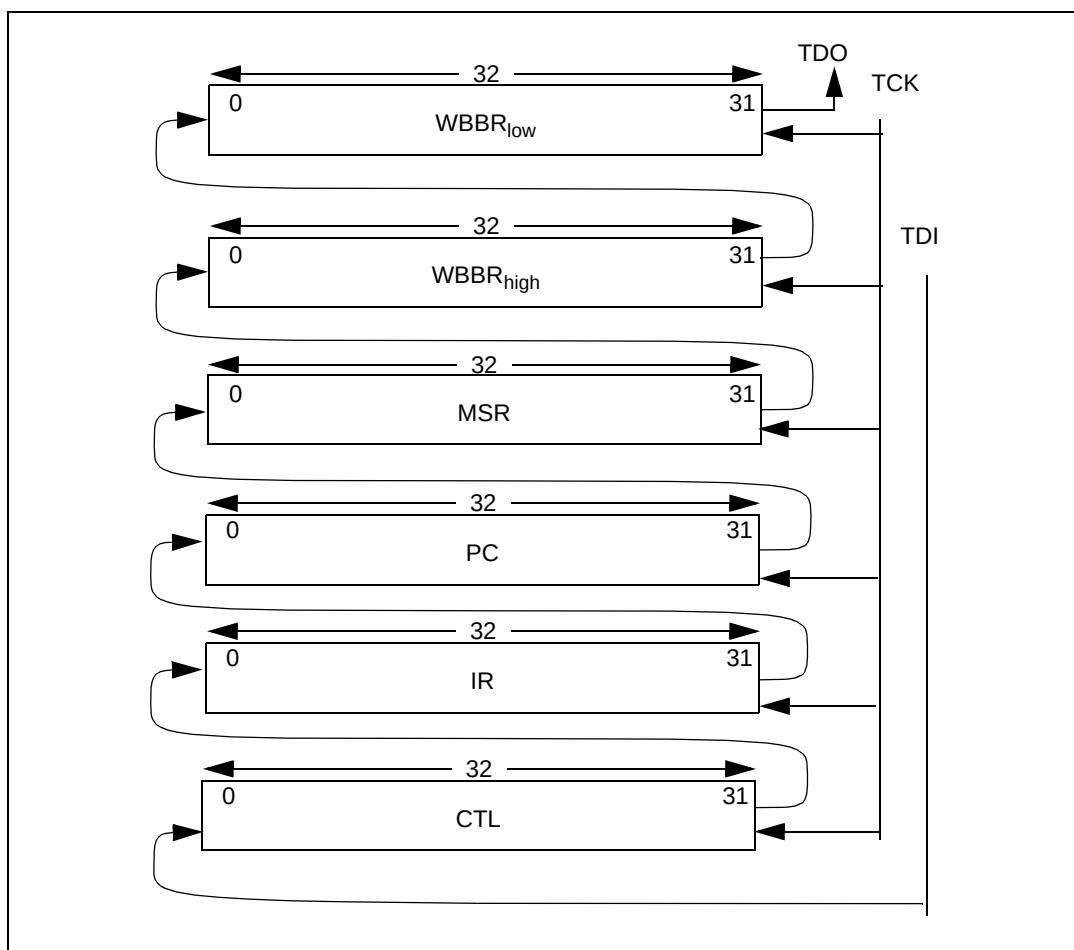


Figure 1582. CPU Scan Chain Register (CPUSCR)

64.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

64.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the

operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

Figure 1583. Control State Register (CTL)

*										IRSTAT14	IRSTAT13	IRSTAT12	IRSTAT11	IRSTAT10	WAITING	PCOFST				PCINV	FFRA	IRSTAT0	IRSTAT1	IRSTAT2	IRSTAT3	IRSTAT4	IRSTAT5	IRSTAT6	IRSTAT7	IRSTAT8	IRSTAT9
										10	11	12	13	14	15					16	17	18	19	20	21	22	23	24	25	26	27

Table 1483. CTL field descriptions

Bit	Name	Description
0:9	*	Internal State Bits These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See Section 64.5.5.3, OnCE Debug output (jd_debug_b) , which is not part of the normal program execution flow, these bits should be set to a 0.
10	IRSTAT14	IR Status Bit 14 This control bit indicates an MPU Instruction Address Compare event status for the IR. 0 No MPU Instruction Address Compare event occurred on the fetch of this instruction. 1 An MPU Instruction Address Compare event occurred on the fetch of this instruction.
11	IRSTAT13	IR Status Bit 13 This control bit indicates an Instruction Address Compare 8 event status for the IR. 0 No Instruction Address Compare 8 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 8 event occurred on the fetch of this instruction.
12	IRSTAT12	IR Status Bit 12 This control bit indicates an Instruction Address Compare 7 event status for the IR. 0 No Instruction Address Compare 7 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 7 event occurred on the fetch of this instruction.
13	IRSTAT11	IR Status Bit 11 This control bit indicates an Instruction Address Compare 6 event status for the IR. 0 No Instruction Address Compare 6 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 6 event occurred on the fetch of this instruction.
14	IRSTAT10	IR Status Bit 10 This control bit indicates an Instruction Address Compare 5 event status for the IR. 0 No Instruction Address Compare 5 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 5 event occurred on the fetch of this instruction.

Table 1483. CTL field descriptions (continued)

Bit	Name	Description
15	WAITING	WAITING State Status This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit. 0 CPU was not in the waiting state when debug mode was entered 1 CPU was in the waiting state when debug mode was entered
16:19	PCOFST	PC Offset Field This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV, which overrides this field). 0000 No correction required. 0001 Subtract 0x04 from PC. 0010 Subtract 0x08 from PC. 0011 Subtract 0x0C from PC. 0100 Subtract 0x10 from PC. 0101 Subtract 0x14 from PC. All other encodings are reserved
20	PCINV	PC and IR Invalid Status Bit This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered. 0 No error condition exists. 1 Error condition exists. PC and IR are corrupted.
21	FFRA	Feed Forward RA Operand Bit This control bit causes the content of the WBBR_{lo} to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR. For most LSP instructions using rA rB as a 64-bit source operand, WBBR_{hi, lo} is used to supply the 64-bit source value. This allows the debug firmware to update processor registers — initialize the WBBR_{lo} with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register. Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent. 0 No action. 1 Content of WBBR_{lo} or WBBR_{hi,lo} used as operand value.

Table 1483. CTL field descriptions (continued)

Bit	Name	Description
22	IRSTAT0	IR Status Bit 0 This control bit indicates a TEA status for the IR. 0 No TEA occurred on the fetch of this instruction. 1 TEA occurred on the fetch of this instruction.
23	IRSTAT1	IR Status Bit 1 This control bit is reserved.
24	IRSTAT2	IR Status Bit 2 This control bit indicates an Instruction Address Compare 1 event status for the IR. 0 No Instruction Address Compare event 1 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 1 occurred on the fetch of this instruction.
25	IRSTAT3	IR Status Bit 3 This control bit indicates an Instruction Address Compare 2 event status for the IR. 0 No Instruction Address Compare 2 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 2 event occurred on the fetch of this instruction.
26	IRSTAT4	IR Status Bit 4 This control bit indicates an Instruction Address Compare 3 event status for the IR. 0 No Instruction Address Compare event 3 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 3 occurred on the fetch of this instruction.
27	IRSTAT5	IR Status Bit 5 This control bit indicates an Instruction Address Compare 4 event status for the IR. 0 No Instruction Address Compare 4 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 4 event occurred on the fetch of this instruction.
28	IRSTAT6	IR Status Bit 6 This control bit indicates a Parity Error status for the IR. 0 No Parity Error occurred on the fetch of this instruction. 1 Parity Error occurred on the fetch of this instruction from the I-Cache or the IMEM.
29	IRSTAT7	IR Status Bit 7 This control bit indicates a Precise External Termination Error status for the IR. 0 No Precise External Termination Error occurred on the fetch of this instruction. 1 A Precise External Termination Error occurred on the fetch of this instruction.
30	IRSTAT8	IR Status Bit 8 This control bit indicates the PowerISA VLE status for the IR. 0 IR contains a BookE instruction. (unused encoding) 1 IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit. <i>Note: This bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).</i>
31	IRSTAT9	IR Status Bit 9 This control bit is reserved.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the

desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit that was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC + 4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events that may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be reentered prior to any further instruction execution.

64.5.9.3 Program Counter (PC) register

The PC is a 32-bit register that stores the value of the program counter that was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

64.5.9.4 Write-Back Bus Register (WBBR_{low}, WBBR_{high})

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. WBBR_{low} holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. For LSP instructions that generate 64-bit results, WBBR_{low} holds the low-order 32 bits of the result. WBBR_{high} holds the updated effective address calculated by a load with update instruction. For LSP instructions that generate 64-bit results, WBBR_{high} holds the high-order 32 bits of the result. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into WBBR_{low}. The contents of WBBR_{low} can then be delivered serially to the external command controller. To update a

processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed, which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the WBBR_{low} to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to [Section 64.5.9.2, Control State Register \(CTL\)](#) for more detail on the CTL_{FFRA} bit.)

WBBR_{low} and WBBR_{high} are generally undefined on instructions that do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

64.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used. This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

64.5.9.6 Exiting Debug mode and interrupt blocking

When exiting debug mode with a Go+Exit, “asynchronous” interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will reenter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

64.5.10 Reserved registers (Reserved)

The reserved registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

64.6 Watchpoint support

e200z425n3 supports the generation and signalling of watchpoints when operating in internal debug mode (DBCR0_{IDM}=1) or in external debug mode (EDBCR0_{EDM}=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with EDBCR0_{EDM} or DBCR0_{IDM}.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVENT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based

events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31]**) for various development control purposes. (See the “Watchpoint Trace Messaging” section in the Core (e200z425n3) Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

[Table 1484](#) shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 1484. Watchpoint output signal assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status

Table 1484. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever p_pm_event transitions from a '0' to a '1' while m_clk is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on PMLCa0 _{PMP}
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on PMLCa1 _{PMP}

Table 1484. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on PMLCa2 _{PMP}
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on PMLCa3 _{PMP}
jd_watchpt[27]	MPU	Memory Protection Unit watchpoint Asserted whenever a the MPU generates a debug event based on region descriptor matches with DEBUG control enabled.
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

64.7 MPU operation during debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR ([Section 64.5.6.3, OnCE Control Register \(OCR\)](#)) for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR ([Section 64.5.6.3, OnCE Control Register \(OCR\)](#)) for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

64.8 Cache array access during debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

64.9 Basic steps for enabling, using, and exiting External Debug mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug mode and initializing debug registers:

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR} , OCR_{WKUP} are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set.

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in [Section 64.5.9.2, Control State Register \(CTL\)](#)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with “no-register selected”, Go cleared, and determines that the PCU has reentered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in [Section 64.5.9.2, Control State Register \(CTL\)](#)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the EDBCR0_{EDM} bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should rewrite the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in [Section 64.5.9.2, Control State Register \(CTL\)](#)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note: These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

65 Debug and Calibration Interface (DCI)

65.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

To provide high-speed calibration capability, the DCI also provides a mechanism to use 5-pin LFAST debug tool which shares its pins with the standard JTAG interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features for the Production device (PD) only (information about the Buddy device (BD) is not included).

65.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- Port sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

65.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Production device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using LFAST or software debug mode.
 - When using LFAST with a connected calibration/debug tool, JTAG packets are received from the tool via LFAST and output JTAG packets are transmitted back to the tool. The LFAST's 5-pin interface consists of an LVDS pair for transmit signals, an LVDS pair of receive signals and a reference clock output signal.
 - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode. The LFAST-based debug interface shares its five pins with the standard JTAG pins.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

As the LFAST-based debug interface shares its five pins with JTAG pins, the DCI provides a safe switching logic to switch to LFAST-based debug mode.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as GTM/SPU, etc. The DCI also provides the Event Out (EVTO) signals to the debug tool in case of the occurrence of any internal debug event.

Figure 1584 shows the DCI block diagram.

Figure 1584. DCI block diagram

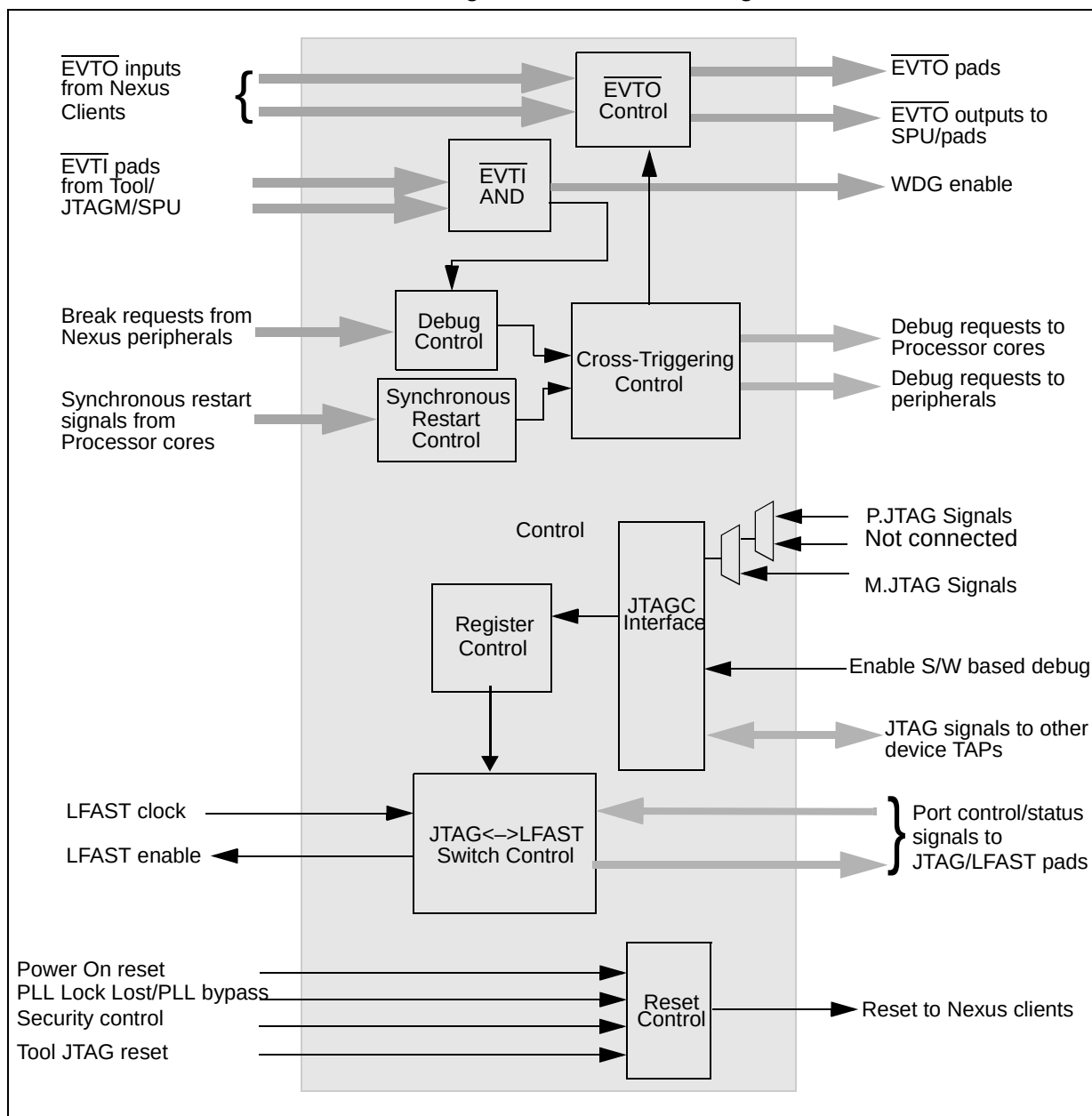
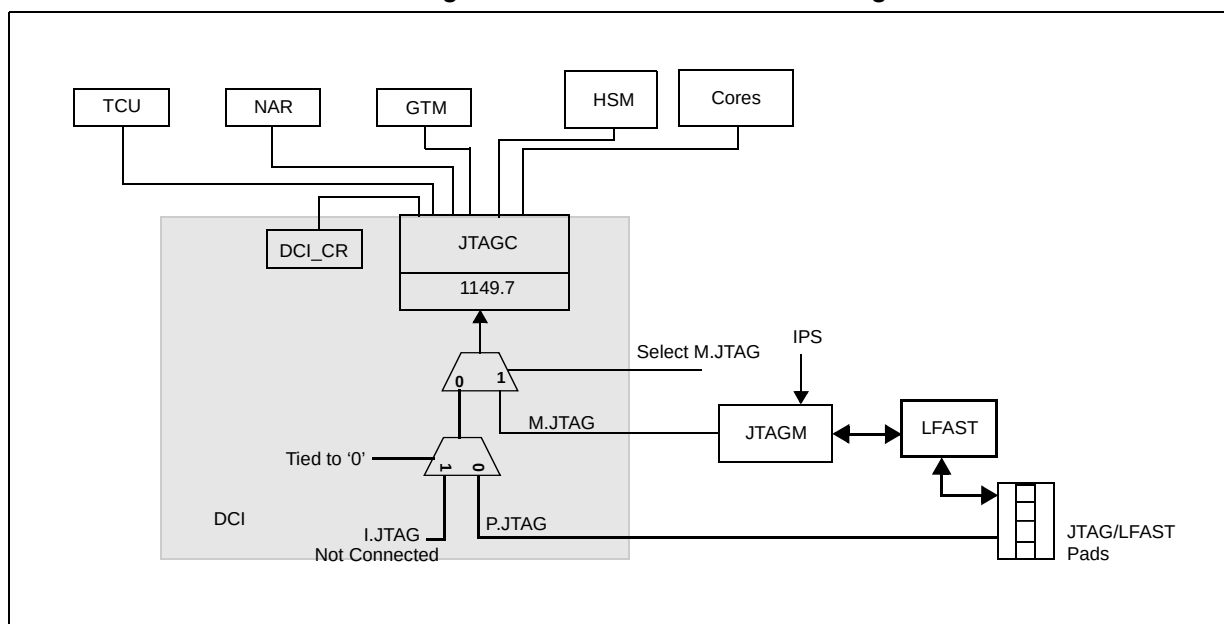


Figure 1585 shows the DCI configuration for the standard production package.

Figure 1585. DCI in a standard configuration



65.1.3 Operating modes

Selection between various JTAG sources is done using a JTAG register bit, DCI_CR[EN_LFAST], and a JTAGM register bit, JTAGM_MCR[DTM]. The JTAG bit, DCI_CR[EN_LFAST], can only be programmed using JTAG instructions and is used to select standard JTAG or LFAST debug tool. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when either DCI_CR[EN_LFAST] or JTAGM_MCR[DTM] is set.

[Table 1485](#) shows a summary of the configuration options.

Table 1485. DCI JTAG source selection

DCI_CR[EN_LFAST]	JTAGM_MCR[DTM]	DCI Source	JCOMP
0	0	P.JTAG (Standard JTAG)	Asserted
1	x	M.JTAG (LFAST)	Asserted
x	1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated.

The M.JTAG provides an alternate interface to the debug logic.

When enabled, it allows high speed debug through the LFAST and JTAGM modules. The high speed link uses LFAST commands that are translated to JTAG commands in the

JTAGM module. The selection between LFAST based debug or software based debug is managed in the JTAGM. However, it is important to note that the dynamic switching between tool-based debug and software debug is not allowed. Software based debug is done only in those cases when an external tool is not present.

An LFAST based tool starts in JTAG mode and then switches to LFAST mode. This switching is managed through a control set of operations which is described in [Section 65.1.3.1, Port sharing between JTAG and LFAST modes](#). During the switch from JTAG to LFAST and back, the DCI uses a latched value of JCOMP to prevent all debug configuration reset.

The LFAST uses LVDS ports for LFAST communication and the JTAG ports uses CMOS level GPIOs. These signals are double-bonded to share the same pins at the tool interface. [Table 1486](#) shows the port mapping.

When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

Table 1486. DCI LFAST/JTAG port sharing

JTAG pad	Type in JTAG mode	LFAST pad	Type in LFAST mode
TDI	Input	LVDS TxN	Output
TMS	Input	LVDS TxP	Output
TDO	Output	LVDS RxP	Input
JCOMP	Input	LVDS RxN	Input
TCK	Input	DRCLK	Output

[Figure 1586](#) shows the DCI mode switching operation.

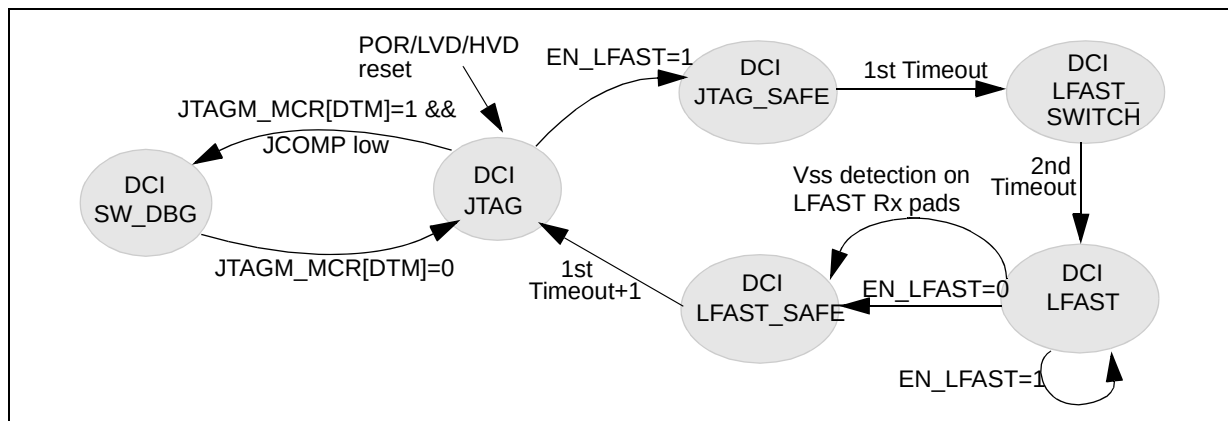


Figure 1586. DCI operating modes state diagram

Note: `.JTAGM_MCR[jtagm_JCOMP]` should not be asserted in LFAST mode to avoid any undesired behavior.

[Table 1487](#) contains the operating mode descriptions.

Table 1487. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.
DCI_JTAG_SAFE	SAFE_JTAG state. When an external tool sets the DCI_CR[EN_LFAST] bit, this indicates that the external tool is going to switch to the LFAST mode. The external tool puts the DCI in Run-Test-Idle state to start the switching operation. To enable safe switching operation, the DCI enters the JTAG_SAFE state where it performs the following operations: – JCOMP is latched inside to control until switching gets completed – The DCI releases control of the JTAG ports and uses high on TMS inside, thus keeping the DCI in the Run-Test-Idle state – An internal counter starts to count to a tool specified value as programmed in DCI_CR[SWITCH_CNTR1].
DCI_LFAST_SWITCH	LFAST port switching state. In this state, the DCI asserts the enable signal to indicate to the LFAST that it is now ready for LFAST port switching. In this mode, the LFAST Rx pads are enabled. The switch counter starts another count operation to count to the DCI_CR[SWITCH_CNTR2] value.
DCI_LFAST	LFAST mode with escape mode enabled. This mode indicates that the switching is now completed and the debug tool is operating in the LFAST mode. However, there is another feature called Escape Mode, which enables the LFAST pads to detect any disconnection fault on its Rx pads. When these pads are working in Escape Mode, a sensing logic is enabled in the pad which detects and flags any out of common mode voltage range for the Rx pads. If this flag is asserted, it is used to force the DCI to go to LFAST_SAFE mode to ensure safe switching to the JTAG state.
DCI_LFAST_SAFE	LFAST_SAFE mode. While the tool is switching from LFAST mode to JTAG mode, it resets the DCI_CR[EN_LFAST] bit. The external tool puts the DCI in the Run-Test-Idle state to start the switching operation. To account for the switching time for the pads, the DCI enters in a safe LFAST state. It then restarts a counter to count to a tool specified value in DCI_CR[SWITCH_CNTR1], and disables both the LFAST function on the pads and the LFAST module.

Figure 1587 shows a block diagram of the JTAG/LFAST connections.

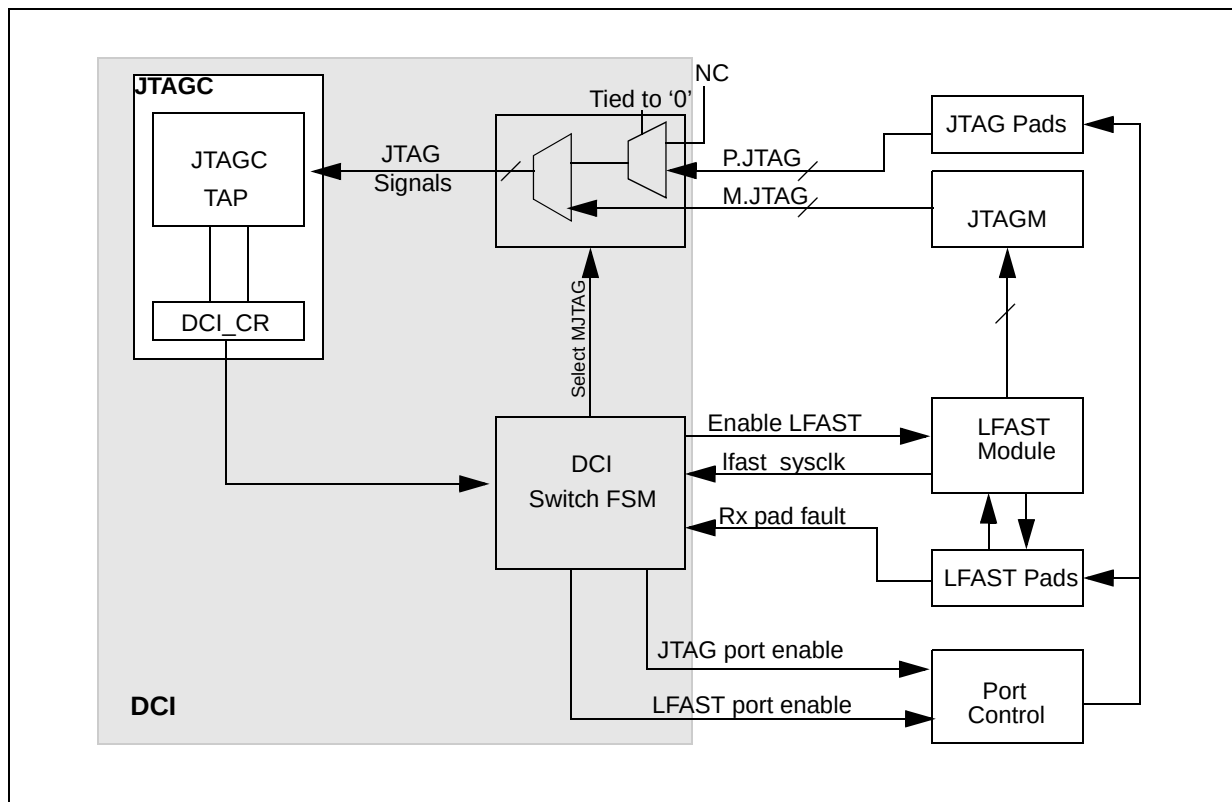


Figure 1587. DCI JTAG/LFAST switch integration

65.1.3.1 Port sharing between JTAG and LFAST modes

By default the DCI is configured in JTAG mode using the standard JTAG port (P.JTAG). Therefore, the initial configuration is done using low speed JTAG signals. In order to switch from JTAG mode to LFAST mode, the tool accesses the following fields in the DCI control register (DCI_CR):

- EN_LFAST—Bit to indicate enable/disable LFAST mode
- SWITCH_CNTR1 and SWITCH_CNTR2—Bit fields to program a count value that corresponds to the delay for switching the pin functions when changing modes.

As the JTAG to LFAST mode switching reflects change in port input/output behavior, it is done through a controlled procedure. The DCI_CR is only accessible through JTAG sequences. While switching from JTAG to LFAST mode, the tool programs this register via P.JTAG to enable switching to LFAST mode. As the pins are shared between JTAG and LFAST functions, a programmable delay based on switch counters is required which allows a safe switching for the MCU as well as for the tool to change its mode. In LFAST mode, the tool can write this register to switch back from high speed debug to low speed debug. Again, the similar switching delay mechanism is used. The switching schematic is shown in [Figure 1588](#).

A programmable switching delay and tri-stating port pins during switching allow a connected tool to safely tune the device turn-around of pin functions from JTAG mode to LFAST mode and vice versa. These delays are programmed according to the time required by the tool to change its operation from JTAG mode to LFAST and vice versa. This reduces the possibility of device damage during switching.

The programmable counter uses the LFAST system clock as the clock source.

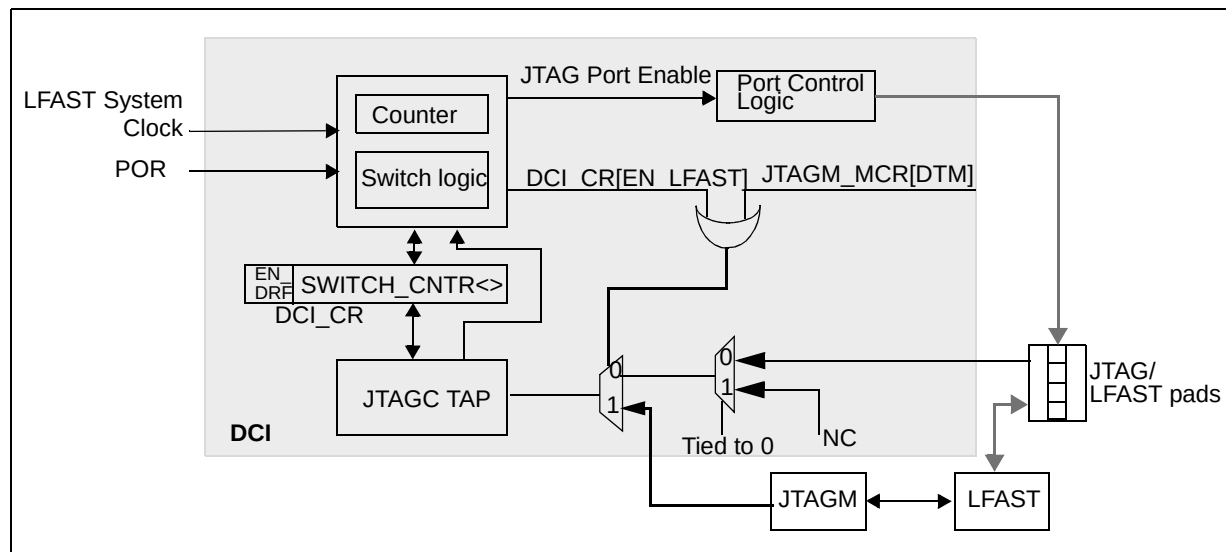


Figure 1588. DCI switching between JTAG and LFAST modes

65.1.3.1.1 Switching from JTAG mode to LFAST mode

The steps to switch operation from JTAG to LFAST mode are as follows:

1. System starts in port JTAG mode after power-on-reset.
2. External tool programs the following bits in the DCI_CR:
 - EN_LFAST
 - SWITCH_CNTR1
 - SWITCH_CNTR2
3. Tool puts JTAG TAP in the Run-Test-Idle mode.
4. An internal finite state machine (FSM) starts the switching operation using the LFAST reference clock. The switching process takes four LFAST reference clocks to start, and an external tool should not change the JTAG signals during this period. The DCI forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. The DCI also triggers a switching counter to allow for some time for the tool to change its mode from JTAG to LFAST.
5. After the counter counts to the DCI_CR[SWITCH_CNTR1] value, the LFAST IP and LFAST pads are enabled. Then the counter is reset. Also, internally JTAGM is selected as the JTAG source for the DCI.
6. Counter counts to the DCI_CR[SWITCH_CNTR2] value to enable the Escape mode feature for the LFAST Rx pads. This time allows the tool to settle down in LFAST mode.
7. Tool starts LFAST transmission when it detects pulses on the DRCLK pin. Tool now can use the LFAST to program the JTAGM to generate JTAG sequences.

Figure 1589 shows the timing sequence.

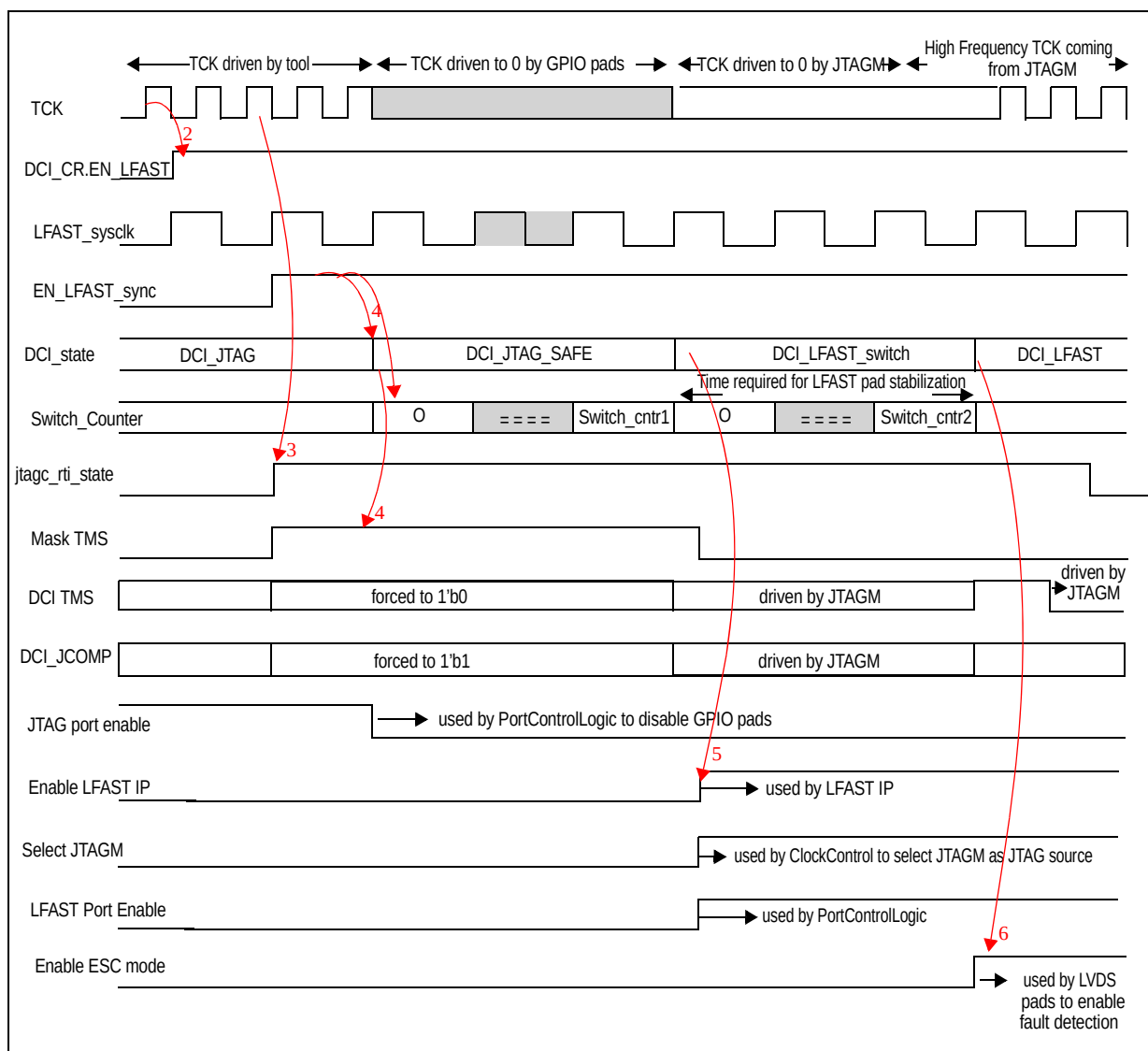


Figure 1589. DCI JTAG to LFAST switching diagram

65.1.3.1.2 Switching from LFAST mode to JTAG mode

The steps to switch operation from LFAST to JTAG mode are as follows:

1. External tool programs the DCI_CR to reset the EN_LFAST bit and SWITCH_CNTR1 for the switching delay.
2. Tool puts JTAG TAP in the Run-Test-Idle mode.
3. An internal FSM starts switching operation. It forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. It also triggers a switching counter to count to DCI_CR[SWITCH_CNTR1] to allow for the tool to change its mode from LFAST to JTAG.
4. After the counter overflows, the JTAG pads are enabled and JTAGM releases its control for the DCI JTAG source.

[Table 1491](#) and [Table 1493](#) summarize the MCU and tool behaviors. [Figure 1590](#) shows the timing sequence.

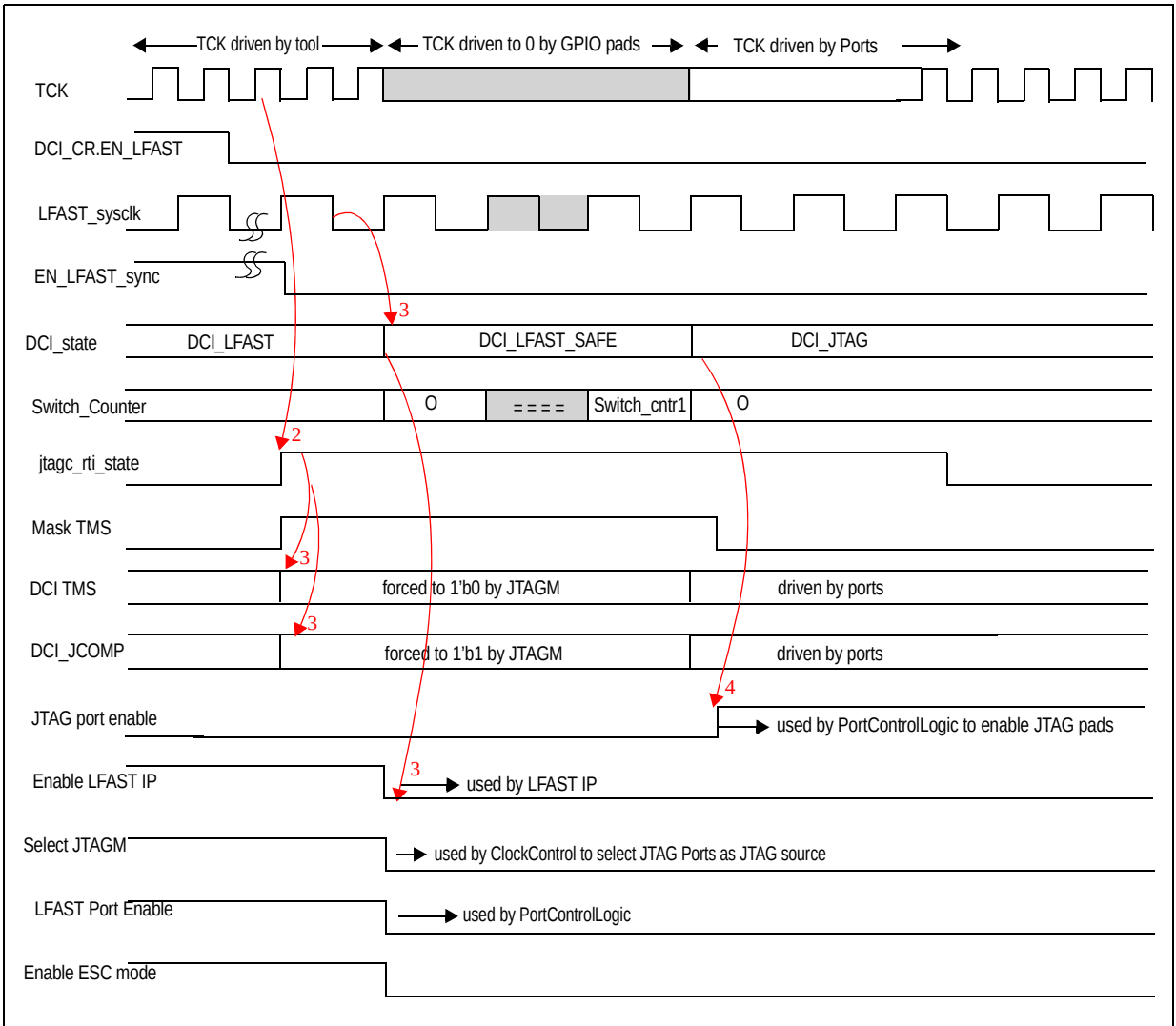


Figure 1590. DCI LFAST to JTAG switching diagram

65.1.3.2 VSS detection

When the LFAST pads are operating in Escape mode, a sensing logic is enabled in the pad which detects and flags any disconnection of the LFAST Rx pads. This flag, if asserted, is used to force the DCI to go to JTAG mode to ensure safe operation. This flag, when asserted, also resets the JTAG TAP to restart in JTAG mode and the EN_LFAST bit is reset accordingly. This flag resets only JTAG TAPs and does not affect any Nexus blocks.

This feature is enabled only when DCI_CR[EN_ESC_MODE] is set. By default this feature is not enabled. To tune the tool behavior it is possible to emulate the pad fault using the DCI_CR[FLVDS_PAD_FLT] bit.

65.1.3.3 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool or LFAST-based tool, it is possible to generate JTAG sequences through software programming using JTAGM IP. The JTAGM, which generates JTAG sequences using LFAST protocol, is also used to generate JTAG sequences using software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

There is a software programmable bit, MCR[jtagm_JCOMP], present in JTAGM block which, when programmed to 0, asserts reset for the JTAGC block, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

Note: .It is recommended to utilize either hardware based debug (JTAG or LFAST) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

If both debug methods are used simultaneously, then an external tool may arbitrarily connect when the SoC has already been halted using the software debug mode. In this case, two consecutive assertions of external JCOMP are necessary to allow the external tool to take control of debug session.

65.1.3.4 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect
- JCOMP
- SoC Debug_Enable signal
- System PLL is out of lock state and not bypassed

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the Nexus reset signal remains asserted until the system clock achieves lock or in bypass mode, as indicated by the system clock lock status signal, and Debug_Enable is asserted and JCOMP is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset.

This reset signal is unaffected by other sources of reset. If Debug_Enable is de-asserted by the SoC security modules (HSM or PASS), then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control, and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 1488. Nexus reset truth table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

65.1.3.5 System watchdog control

The DCI is also used to enable or disable the Software Watchdog Timer (SWT). The DCI provides a single bit SWT enable signal which is used to control all SWTs in the MCU. For this, the EVTI[0] pin is used. If the EVTI[0] pin remains high when the tool asserts JCOMP, then the SWT is enabled. But if the tool drives the EVTI[0] pin low before asserting JCOMP pin, then the SWT is disabled. By default the SWT is enabled and a tool has an option to enable/disable the SWT using the EVTI[0] pin.

65.1.3.6 Break control

The DCI provides central break control among the CPUs and non-CPU debug modules such as GTM, DMA, Watchdog and other peripherals.

There are various different input sources that provide debug requests:

- [BLG_DCI_0016][Covers: ADD23.025][Covers: ADD23.029][Covers: ADD23.134]EVTI[1:0] signals from EVTI pads[end]
- EVTI[1:0] signals from JTAGM
- EVTI[1:0] signals from the Sequence Processing Unit (SPU)

The DCI generates following debug request signals:

- Debug Request signal to be used for all non-CPU peripherals
- External Debug Request to be used to request external debug event to processor cores
- OnCE Debug Request to be used to request OnCE debug event to processor coresAll break control requests are issued simultaneously to CPUs and peripherals, then synchronized to each clock domain

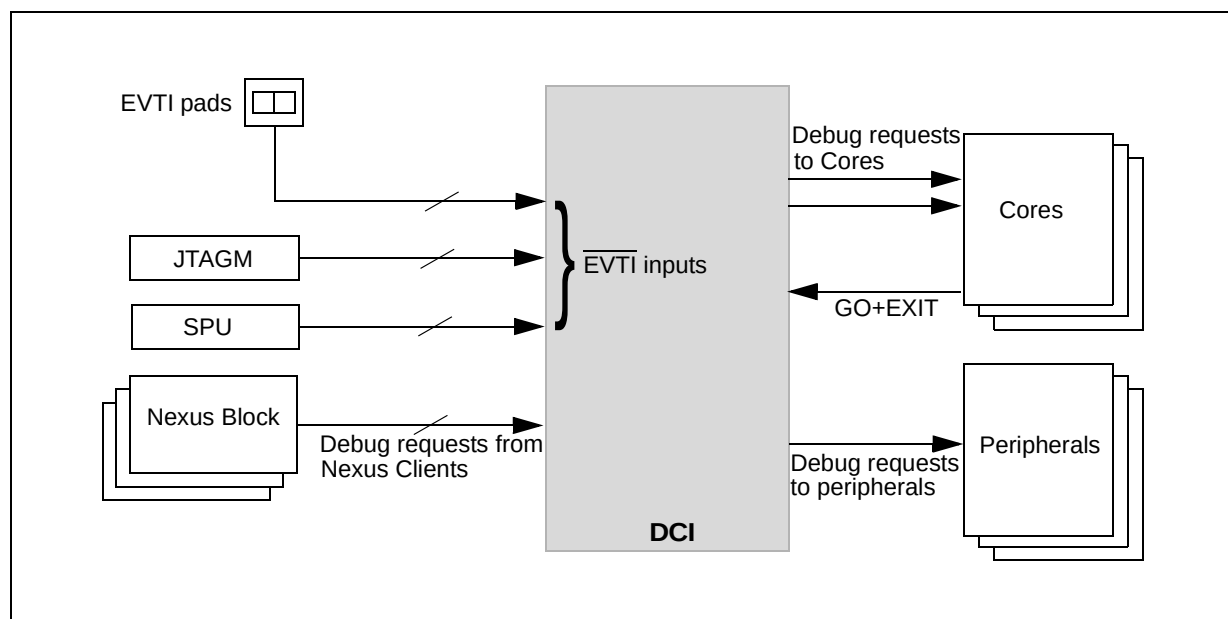


Figure 1591. DCI debug control mechanism

The various sources of $\overline{\text{EVTI}}[1:0]$ inputs are synchronized and logically ANDed in the DCI to generate two debug requests $\text{DCI_EVTI}[1:0]$ for the processor cores and non-CPU peripherals. Of these two inputs, $\text{EVTI}[0]$ is used to generate debug requests for non-CPU peripherals and for processor cores through the external debug request signal. The other input, $\text{EVTI}[1]$, is also used to generate debug requests for non-CPU peripherals and it generates debug requests to processor cores through the OnCE debug request signal.

The external debug request to processor core is asserted as a single cycle pulse and is synchronized on the respective core clock. The OnCE debug request is negated when the acknowledge signal from the Cores is asserted. The CPU(s) remain in the debug state until GO+EX command is executed for that CPU. Exiting a CPU out of debug state by executing GO+EX command does not take other CPUs out of debug state. The peripheral debug request is negated when the processor core(s) execute GO+EXIT command. The $\text{EVTI}[1:0]$ signals coming from the pads are enabled for debug control using $\text{DCI_CR}[5:4]$ bits.

In order to selectively enable/disable debug requests to system peripherals, the respective debug enable bit inside the peripheral should be used.

[Figure 1592](#) shows the timing sequence.

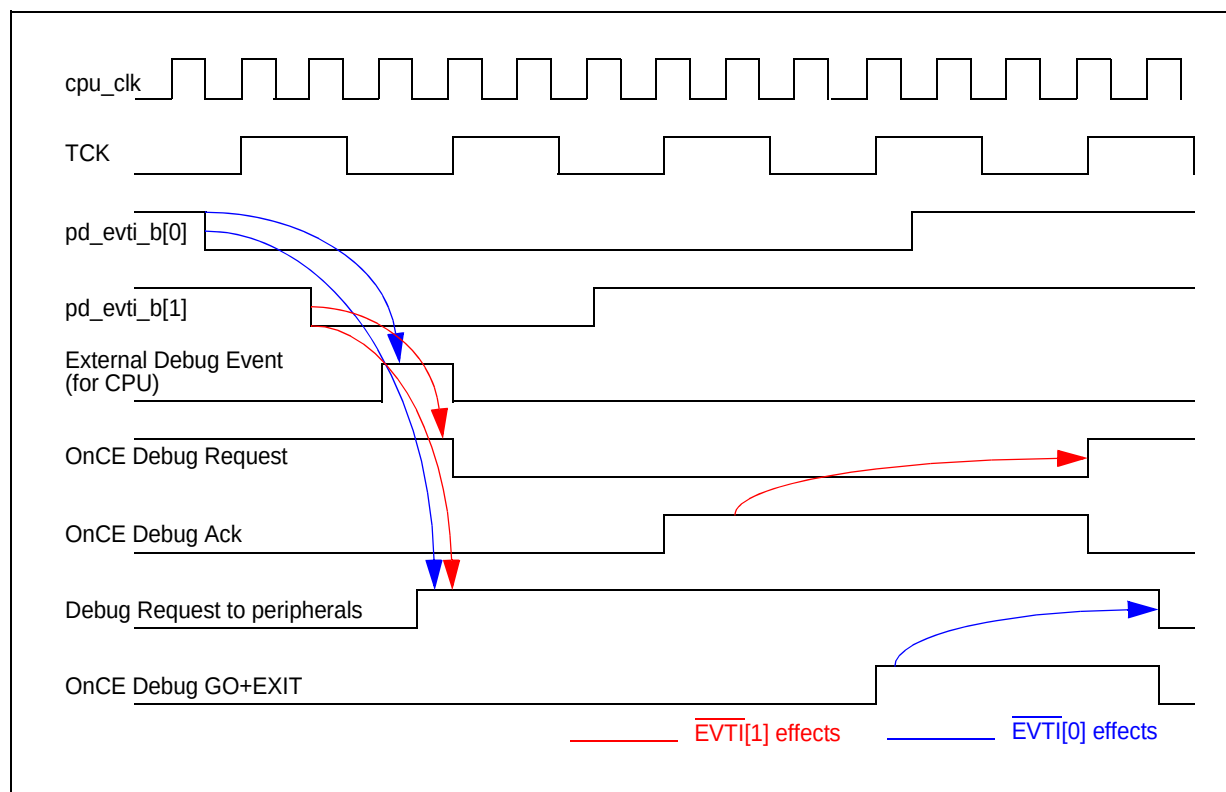


Figure 1592. DCI debug signal generation

65.1.3.7 Cross-triggering control

Many of the blocks in the device, including the Nexus trace blocks and more sophisticated peripherals, have debug features that allow them to request device-wide breaks to enter debug mode. Many blocks can be programmed to halt operation upon debug mode entry or ignore it and continue normal operation. All debug mode entry requests are input to the DCI, ORed together and the result is used to produce the debug requests to the CPUs and non-CPU peripherals.

The cross-triggering feature is implemented through selectively enabling debug control bits in the respective modules. Processor cores (enabled using DBCR0[DEVT1] bit) and non-core peripherals can be programmed to halt operation through their respective control bits. Cross-triggering provides a selective feature to put modules in debug mode upon receiving halt request signals from Nexus modules.

65.1.3.8 Synchronous restart control

The DCI implements a synchronous restart control feature to synchronously restart all debug peripherals (CPU and non-CPU peripherals which are put in debug state). This is accomplished with the following steps:

1. The tool sequentially scans into CPUSCR for each processor core to restore the proper processor state.
2. The tool scans into the Production device tap controller the AUX_PARALLEL instruction, which then selects all processor core tap controllers in parallel.
3. To synchronously exit any combination of processor cores from debug mode, the tool simultaneously scans into all processor core tap instruction registers (OCMD) the 10-bit sequence 0x180 (i.e. set GO and EX, clear RS[0:6]).
4. If a processor core is out of debug mode, it sets a flag OnCE GO+EXIT.
5. When the OR of the processor core OnCE GO+EXIT signals is asserted, then all other selected non-processor cores and peripherals in debug would also be synchronously exited from debug mode. It is done by negating peripheral debug request signal.

65.1.3.9 EVTO management

The EVTO management block diagram is presented in [Figure 1593](#).

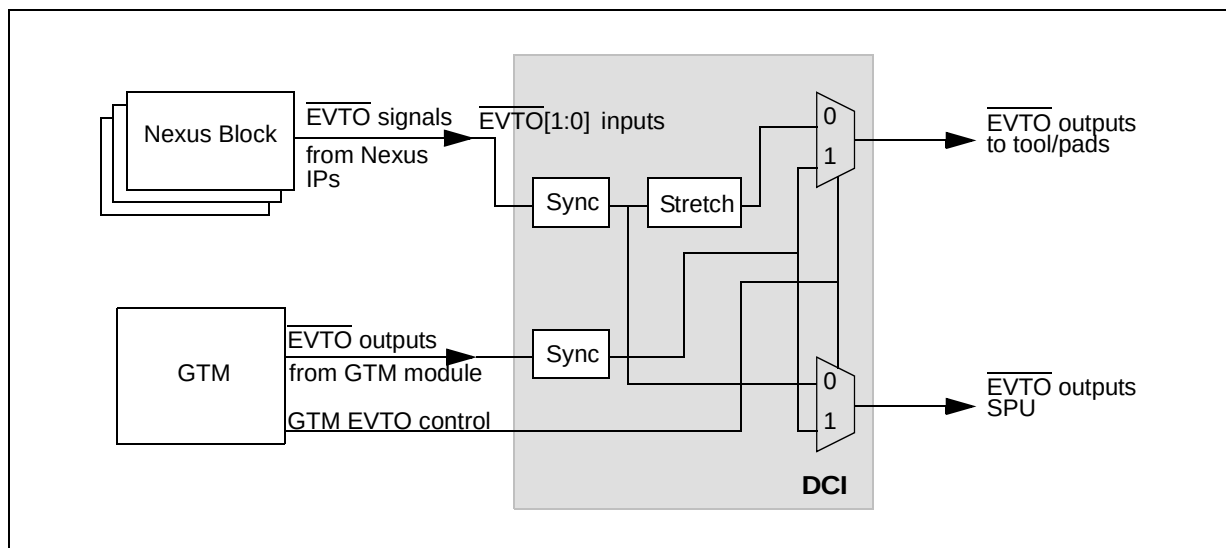


Figure 1593. DCI EVTO management

The DCI module controls sharing of the EVTO outputs between all Nexus Clients including the cores that produce $\overline{\text{EVTO}}[1:0]$ signals. The DCI uses a high frequency clock to sample EVTO signals coming from all modules which might generate EVTO outputs at their own clock frequency. After receiving a single clock period of asserted $\overline{\text{EVTO}}[1:0]$ signals from any Nexus client, the DCI immediately asserts $\overline{\text{EVTO}}[1:0]$ outputs and stretches it for a minimum of one output clock period for the output to the tool ([Figure 1594](#)). The DCI also generates EVTO outputs in other device clock domains so that other Nexus clients can receive EVTO inputs without the need of any synchronization. EVTO function is active as long as the DCI is not in reset.

The DCI also allows the GTM to take control of the EVTO outputs by controlling the EOS0/EOS1 fields of the GTMDI_DC register. If these fields are programmed to HALT/TIM/TOM/ATOMEVTO value, the GTM EVTO outputs are directly available on the ports bypassing the stretch logic.

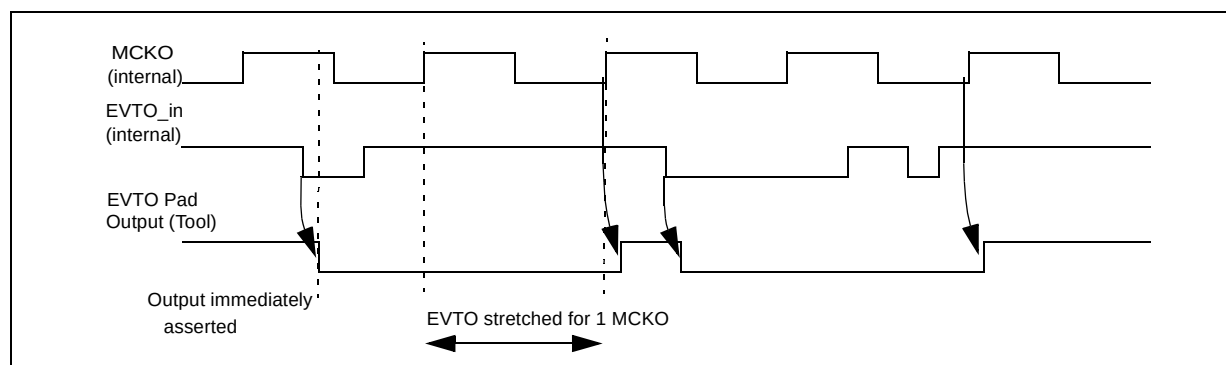


Figure 1594. DCI EVTO timing diagram

65.1.3.10 Reset control

The DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- [BLG_DCI_0038][Covers: ADD23.026][Covers: ADD23.032]Force functional reset—This reset is generated by programming the DCI_CR[FPD_FUNC] bit. When set, it forces functional reset. This bit has to be reset by the tool to negate the reset signal.[end]
- [BLG_DCI_0043][Covers: ADD23.026][Covers: ADD23.032]Force POR or destructive reset—This reset is generated by programming DCI_CR[FPD_DEST] bit. When set it forces destructive reset. This bit has to be reset by the tool to negate the reset signal.[end]

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

65.1.3.11 Security

The SoC security logic provides a Debug_Enable signal to the DCI. The DCI uses the Debug_Enable signal as a control input to generate debug module reset. When debug disable is requested, the DCI keeps all debug blocks in the reset state. This is a protection mode used to disable debug accesses to the peripheral when the device is in a secured state.

65.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

65.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

65.3.1 DCI control register (DCI_CR)

The DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. [Figure 1595](#) shows the format of the DCI_CR.

Figure 1595. DCI control register (DCI_CR)

	31	30	29	28	27	26	25	24
R	SWITCH_CNTR2							
W								
Reset	1	1	1	1	1	1	1	1
	23	22	21	20	19	18	17	16
R	SWITCH_CNTR2							
W								
Reset	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8
R	SWITCH_CNTR1[4:0]				FLVDS_PAD_FLT		FPD_FUN_C	FPD_DEST_RST
W								
Reset	1	1	1	1	1	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved	FORCE_LFAST	EVTI1_PAD_EN	EVTI0_PAD_EN	EVTO1_PAD_EN	EVTO0_PAD_EN	EN_ESC_MODE	EN_LFAST
W								
Reset	0	0	0	0	0	0	0	0

The DCI_CR is described in [Table 1489](#).

Table 1489. DCI_CR field descriptions

Field	Description
31:16 SWITCH_CNTR2	Width of Switch_CNTR2. Decides the second timeout value while switching from JTAG to LFAST mode. The maximum timeout duration = $32,768 \times T_{lfast_sysclk}$
15:11 SWITCH_CNTR1	Width of Switch_CNTR1. Decides the first timeout value while switching from JTAG to LFAST mode, and the timeout value while switching from LFAST mode to JTAG mode. The maximum timeout duration = $32 \times T_{lfast_sysclk}$

Table 1489. DCI_CR field descriptions (continued)

Field	Description
10 FLVDS_PAD_FLT	Force LVDS Pad Fault. Forces the pad fault condition to tune the tool without actually having the pad fault on LFAST Rx pads. This bit is effective only when DCI_CR[EN_ESC_MODE] is set. 0 Do not force Rx LVDS Pad Fault 1 Force Rx LVDS Pad Fault if DCI_CR[EN_ESC_MODE] is set
9 FPD_FUNC	Force Functional reset 0 Do not force functional reset 1 Force functional reset
8 FPD_DEST	Force Destructive reset 0 Do not force destructive reset 1 Force destructive reset
6 FORCE_LFAST	Force enable for LFAST 0 LFAST not forced to enable and is controlled using DCI_CR[EN_LFAST] bit 1 LFAST forcefully enabled (without using DCI_CR[EN_LFAST] bit)
5 EVTI1_PAD_EN	$\overline{\text{EVTI}}[1]$ feature enable 0 $\overline{\text{EVTI}}[1]$ disabled (default) 1 $\overline{\text{EVTI}}[1]$ enabled
4 EVTI0_PAD_EN	$\overline{\text{EVTI}}[0]$ feature enable 0 $\overline{\text{EVTI}}[0]$ disabled (default) 1 $\overline{\text{EVTI}}[0]$ enabled
3 EVTO1_PAD_EN	Output enable for $\overline{\text{EVTO}}[1]$ pad 0 $\overline{\text{EVTO}}[1]$ pad output not enabled (default) 1 $\overline{\text{EVTO}}[1]$ pad output enabled
2 EVTO0_PAD_EN	Output enable for $\overline{\text{EVTO}}[0]$ pad 0 $\overline{\text{EVTO}}[0]$ pad output not enabled (default) 1 $\overline{\text{EVTO}}[0]$ pad output enabled
1 EN_ESC_MODE	Enable Escape Mode. Fault detect feature enable for LFAST Rx pads 0 Fault detect feature not present for LFAST Rx pads (default) 1 Fault detect feature present for LFAST Rx pads and are enabled after the successful switching to LFAST mode
0 EN_LFAST ⁽¹⁾	Enable for LFAST-based tool selection. This bit can be reset either by programming by tool, or pad fault condition in which JTAGC block is reset. 0 LFAST tool not enabled. DCI operates in JTAG mode (default) 1 LFAST tool enabled

1. DCI_CR is a JTAG data register and is not accessible for software read. However, the DCI_CR[EN_LFAST] bit is mapped to JTAGM register bit JTAGM_SR[1] to allow for software read access and a mechanism for software to be aware of an active debug tool interface.

65.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter which describes how the modules are configured.

65.4 Functional description

65.4.1 DCI mode transition

After power-on reset, the DCI starts in JTAG mode. To enable LFAST mode, the external tool has to first operate in JTAG mode, configure DCI and then switch to LFAST mode. This requires certain careful steps so as to avoid any contention on double-bonded pads for JTAG/LFAST. [Table 1490](#) and [Table 1491](#) show the steps and expected states for the MCU and tool for JTAG to LFAST switching. [Table 1492](#) and [Table 1493](#) contains the different steps, and the MCU and tool expected states for the LFAST to JTAG switching.

Table 1490. MCU operation (JTAG to LFAST switching)

MCU pin	JCOMP low	JCOMP high (JTAG mode)	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	1st Timeout	2nd Timeout	LFAST ICLC enable command received
TDI	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxN
TMS	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxP
TDO	Hi-Z	Logic Output	Logic Output	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxP
JCOMP	Logic Input	Logic Input	Logic Input Latch	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxN
TCK	Logic Input	Logic Input	Logic Input	Hi-Z	Logic Output	Logic Output	Logic Output
Mode Escape	Inhibit	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable
LFAST_SYSCLK_EN	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable	Enable

Table 1491. Tool operation (JTAG to LFAST switching)

Tool	JCOMP low	JCOMP high (JTAG mode) ⁽¹⁾	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	TCK detected	Internal timer elapse
TDI	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxN
TMS	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxP
TDO	Hi-Z	Logic Input	Hi-Z	Hi-Z	LVDS Tx	LVDS RxP
JCOMP	Hi-Z	Logic Output (High)	Hi-Z	Hi-Z	LVDS Tx	LVDS RxN
TCK	Don't Care	Logic Output	Logic Input	Logic Input	Logic Input	Logic Input

1. This state is used to set DCI_CR[EN_LFAST] bit.

Table 1492. MCU operation (LFAST to JTAG switching)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
TDI	LVDS Tx	Logic Input	Logic Input
TMS	LVDS Tx	Logic Input	Logic Input
TDO	LVDS Rx	Hi-Z	Logic Output
JCOMP	LVDS Rx	Logic Input Latch	Logic Input (sample JCOMP pin)
TCK	Logic Output	Logic Input	Logic Input
Mode Escape	Enable	Inhibit	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit	Inhibit

Table 1493. Tool operation (LFAST to JTAG switching)

Tool	Send command to clear DCI_CR[EN_LFAST]	LFAST transmits idle after command sent	TCK input clock stops or LVDS Rx both low	Internal timeout: start sending JTAG messages
TDI	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TMS	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TDO	LVDS Tx	Hi-Z	Logic Input	Logic Input
JCOMP	LVDS Tx	Hi-Z	Logic Output (High)	Logic Output (High)
TCK	Logic Input	Logic Input	Logic Output	Logic Output

65.4.2 LFAST LVDS pad fault

It is possible that when the tool is operating in LFAST mode, the tool may get disconnected due to external disturbances, which could put the MCU in an undetermined state. It is imperative to keep the MCU in a safe state. Thus the LFAST Rx pads have a sensing logic which detects if the Rx pad voltages are not equal to the common mode voltage when the DCI is in LFAST mode. If it is found that the LFAST Rx pads are not corresponding to the common mode voltage, the pad logic sends a signal to the DCI which then puts the DCI into JTAG mode. [Table 1494](#) and [Table 1495](#) contain the different steps and expected states for the MCU and tool for LFAST to JTAGRST switching.

Table 1494. MCU operation (LFAST to JTAGRST switching)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
TDI	LVDS TxN	Logic Input
TMS	LVDS TxP	Logic Input
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Input
TCK	Logic Output	Logic Input
Mode Escape	Enable	Inhibit

Table 1494. MCU operation (LFAST to JTAGRST switching) (continued)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
LFAST_SYSCLK_EN	Enable	Inhibit
Internal TRST	Negated	Asserted

Table 1495. Tool operation (LFAST to JTAGRST switching)

Tool	LFAST mode	Tool forces JCOMP low
TDI	LVDS TxN	Don't Care
TMS	LVDS TxP	Don't Care
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Output (Low)
TCK	Logic Input	Don't Care

66 JTAG Controller (JTAGC)

66.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

66.1.1 Block Diagram

The following is a block diagram of the JTAG Controller (JTAGC) block.

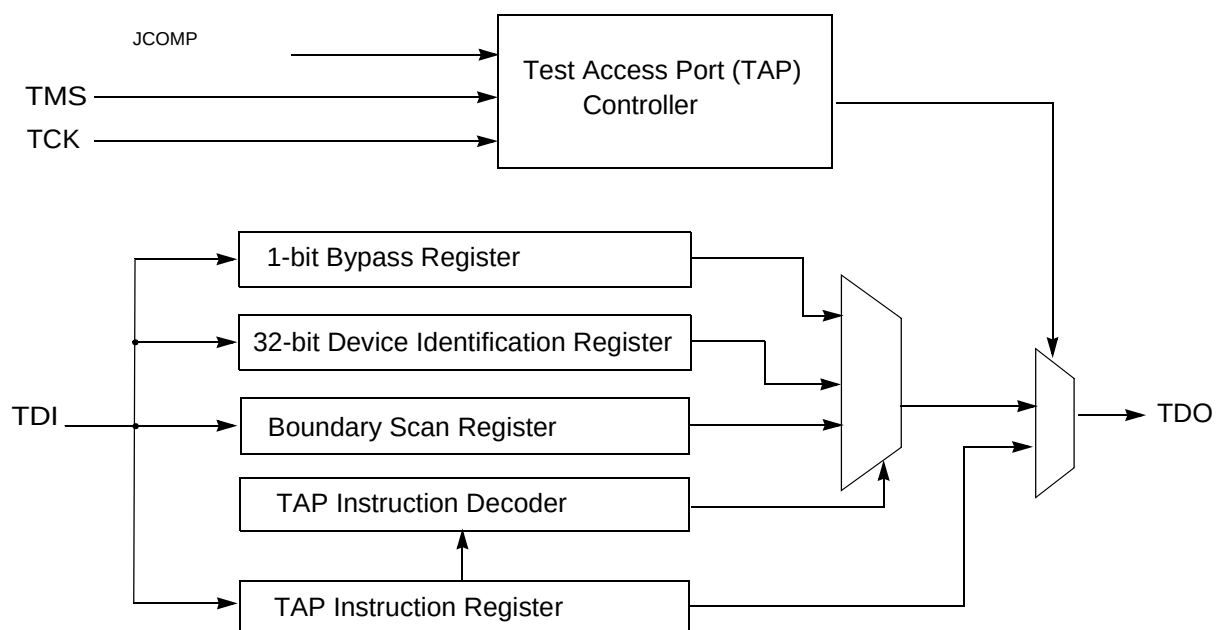


Figure 1596. JTAG (IEEE 1149.1) block diagram

66.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
 - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control and the ability to share the TAP.
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to [Table 1499](#) for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Data registers, bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

66.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

66.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

66.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in [Section 66.4.4, JTAGC block instructions](#).

66.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

66.2 External signal description

The JTAGC consists of five signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 1496. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z ⁽¹⁾	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

66.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

66.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

66.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in [Section 66.4.3, TAP controller state machine](#).

66.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

66.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

66.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

66.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	5	4	3	2	1	0
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

Figure 1597. 6-bit instruction register

66.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

66.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

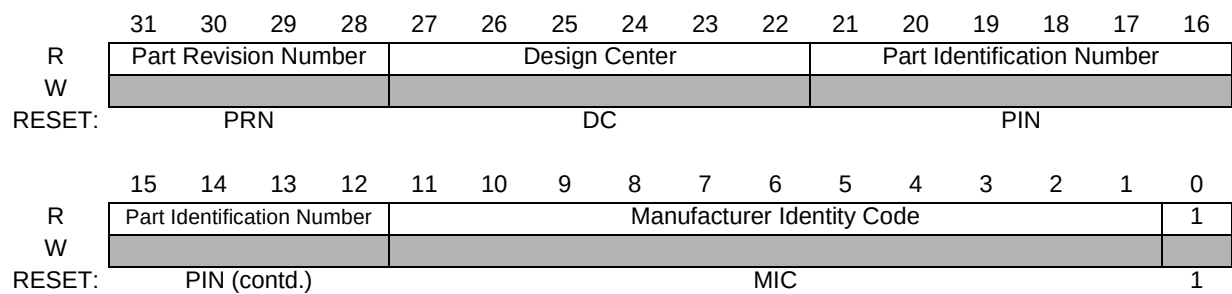


Figure 1598. Device identification register

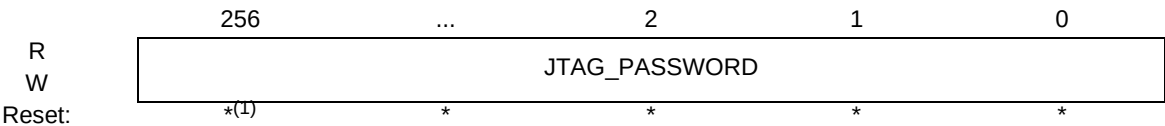
The following table describes the device identification register functions.

Table 1497. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x0.
DC	Design Center. Indicates the design center. Value is 0x2B.
PIN	Part Identification Number. Contains the part number of the device. Value is 0x30A.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x020.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

66.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256'b0.

Figure 1599. JTAG_PASSWORD register

The following table describes the JTAG_PASSWORD register functions.

Table 1498. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

66.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in [Section 66.4.5, Boundary scan](#). The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

66.4 Functional description

This section explains the JTAGC functional description.

66.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

66.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to [Section 66.4.4.8, ACCESS_AUX_x instructions](#).

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

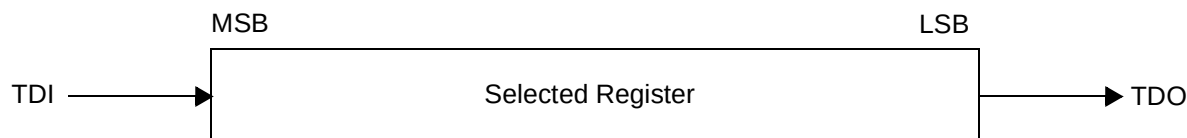
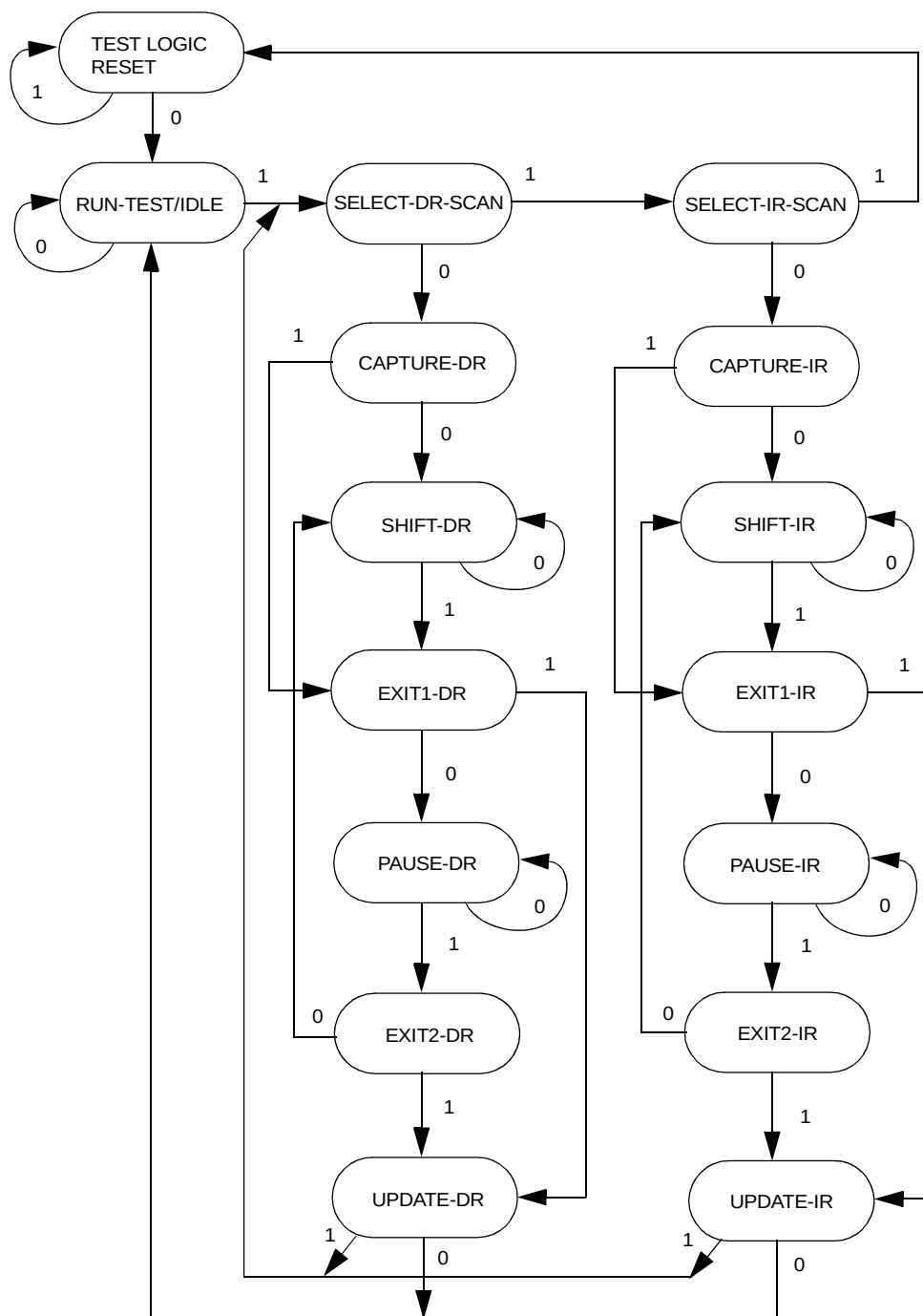


Figure 1600. Shifting data through a register

66.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition represents the value of TMS at the time of a rising edge of TCK.

Figure 1601. IEEE 1149.1-2001 TAP controller finite state machine

66.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

66.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

66.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 1499. 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
Factory debug reserved	000101	Intended for factory debug only
Factory debug reserved	000110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register while three-stating all output pins and asserting functional reset
Factory debug reserved	001010	Intended for factory debug only
CLAMP	001100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ENABLE_DCI_CR	001110	Enables access to DCI CR register
ENABLE_DCI_PINCR	001111	Enables access to DCI PINCR register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
Reserved	100000	Reserved
ACCESS_AUX_NR_PD	100001	Enables access to the NAR TAP controller on PD
Reserved	100010	Reserved
Reserved	100011	Reserved

Table 1499. 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	100100	Reserved
Factory test reserved	100101	Intended for factory test only
ACCESS_AUX_JDC	100110	Enables access to the JDC TAP controller, to access the HSM challenge and response registers
ACCESS_AUX_HSM	100111	Enables access to the HSM core TAP controller
ACCESS_AUX_CORE_0	101000	Enabled access to E200 Core 0
ACCESS_AUX_CORE_1	101001	Enables access to E200 Core 1
ACCESS_AUX_CORE_2	101010	Enables access to E200 Core 2
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
ACCESS_AUX_GTM	110011	Enables access to GTM TAP controller
ACCESS_AUX_BUS_MON_0	110100	Enables access to NXMC Client 0 TAP controller
ACCESS_AUX_BUS_MON_1	110101	Enables access to NXMC Client 1 TAP controller
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
Reserved	111001	Reserved
ACCESS_AUX_SPU_A	111010	Enables access to SPU Client TAP controller
Reserved	111011	Reserved
ACCESS_AUX_PARALLEL ⁽¹⁾	111100	Enables access to all cores in parallel
Reserved	111101	Reserved
Reserved	111110	Reserved
BYPASS	111111	Selects bypass register for data operations
Reserved ⁽²⁾	All other opcodes	Decoded to select bypass register

1. The use of this instruction is limited to the synchronous exit of debug mode via the execution of the GO+EXIT OnCE command. The execution of other instructions or commands is not supported
2. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

66.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

66.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- First, the SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- Secondly, the PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

66.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

66.4.4.4 EXTEST—External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

66.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

66.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

66.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

66.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

66.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

66.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

66.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

67 IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

67.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

67.2 Abbreviations

Table 1500. Abbreviations

Abbreviation	Expansion
APU	Advanced Protocol Unit
APFC	Auxiliary Pin Function Control
CGM	Conditional Group Member
CGMC	Conditional Group Membership Count
CID	Controller ID
CJTAG	Compact JTAG
CLTAPC	Chip-Level TAPC
CP	Check Packet
CPA	Check Process Active
DCU	Data Channel Unit
EOT	End of Transfer
EPU	Extended Protocol Unit
Jscan	JTAG Scan
Mscan	Maximum-flexibility Scan
nTRST	Active low Test Reset
Oscan	Optimized Scan
PSS	Pause Selection State
RSU	Reset and Selection Unit
SP	Scan Packet
SPA	Scan Packet Active
Sscan	Segmented Scan
SSD	Scan Selection Directive
SSM	Scan State Machine
STL	System Test Logic

Table 1500. Abbreviations (continued)

Abbreviation	Expansion
TAP	Test Access Port
TAP.1	An IEEE 1149.1 Test Access Port
TAP.7	An IEEE 1149.7 Test Access Port
TCA	Tap Controller Address
TCK	Test Clock
TCKC	Test Clock Compact
TDI	Test Data Input
TDIC	Test Data Input Compact
TDO	Test Data Output
TDOC	Test Data Output Compact
TMS	Test Mode Select
TMSC	Test Mode Select Compact
TS	Target System
ZBS	Zero Bit Scan

67.3 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backwards compatible with IEEE P1149.1. This standard is also known as “Compact JTAG” or “CJTAG”.

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths
- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

67.3.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- **Compliant**
 - Compliant with the IEEE 1149.1 standard
 - Four-pin operation using the “Standard Protocol” — IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
 - More than one TAPC may be associated with the TAP (once made visible by private instructions)
- **Extended**
 - Compatible with the IEEE 1149.1 standard
 - Four-pin operation using the “Standard Protocol” — IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
 - Additional test/debug functions
- **Advanced**
 - Two-pin operation using the “Advanced Protocol” — signaling where TCK(C) and TMS(C) signals transfer control/data information
 - The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

67.3.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
 - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
 - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
 - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors.
 - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
 - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

67.3.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in [Table 1501](#). Signal names that end in “C” are associated with expanded functionality (above that provided by the IEEE 1149.1 standard).

The notation used in the table is as follows:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the TCKC, TMS, TDIC, TDOC, and nTRST pins.

Table 1501. TAP.7 signal list

Signal Name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
nTRST	Test reset	O	O	O	O	O
Mandatory signal count (minimum)		2	4	4	4	4

67.3.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in [Figure 1602](#). It is described with the hardware layers listed below:

- **STL – System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP.7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

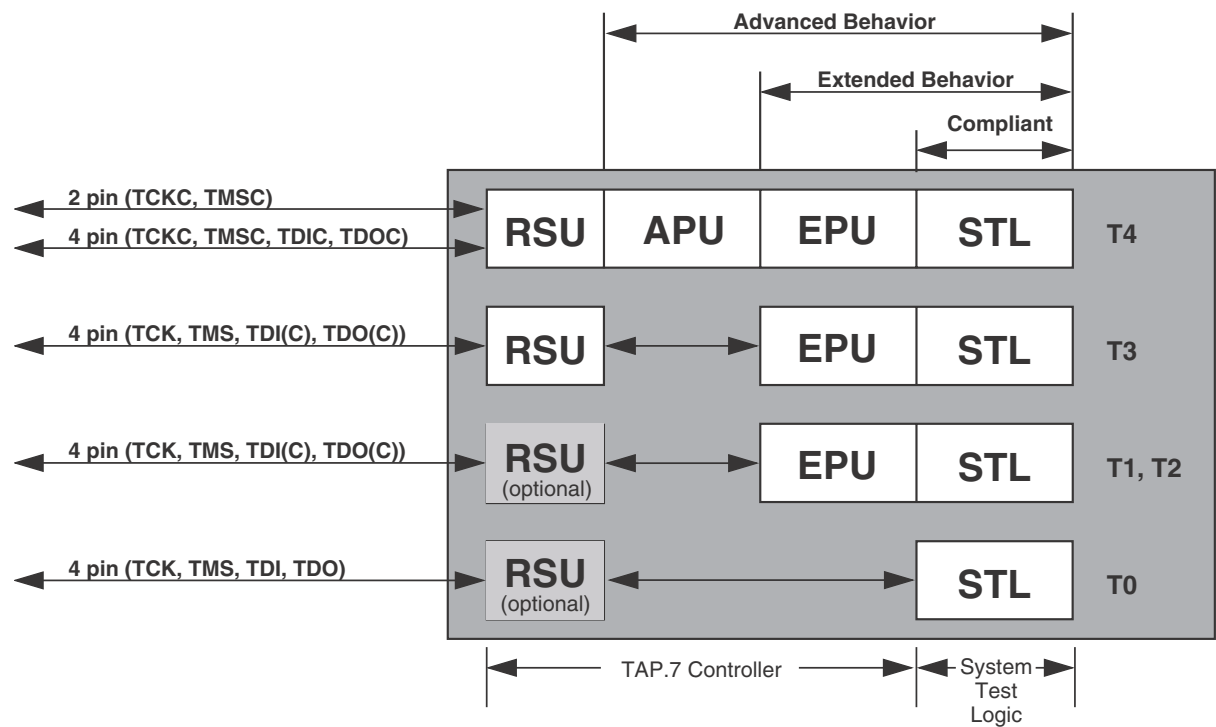


Figure 1602. TAP.7 Controller Architecture

67.3.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in [Figure 1603](#).

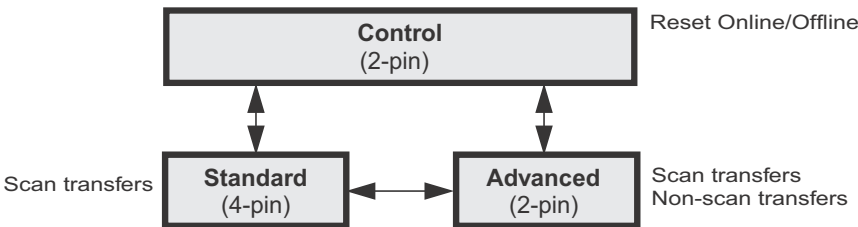


Figure 1603. TAP.7 Behaviors

The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in [Table 1502](#).

Table 1502. Class/protocol relationship

Class	Protocol		
	Standard	Control	Advanced
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

67.3.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

67.3.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMS signaling sequences.

67.3.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

67.3.6 Operating models

[Figure 1604](#) illustrates a simplified view of the CJTAG operating model.

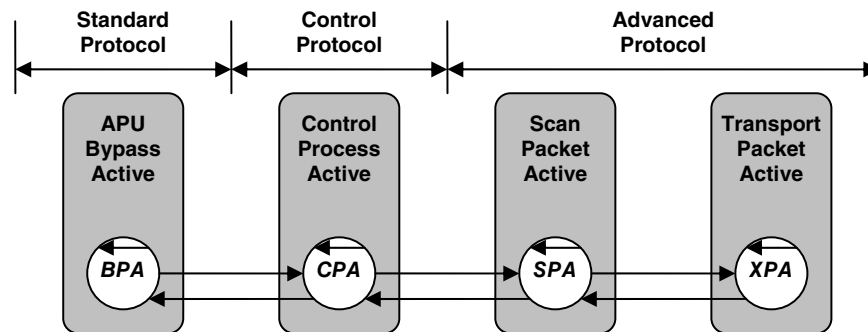


Figure 1604. CJTAG operating model

67.4 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

67.4.1 T0 Functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

67.4.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

67.4.3 T2 functions

- “Super Bypass” function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

67.4.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

67.4.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

67.5 Ancillary services

67.5.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options
- Escape sequences
- TAPC state machine

67.5.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset. (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC

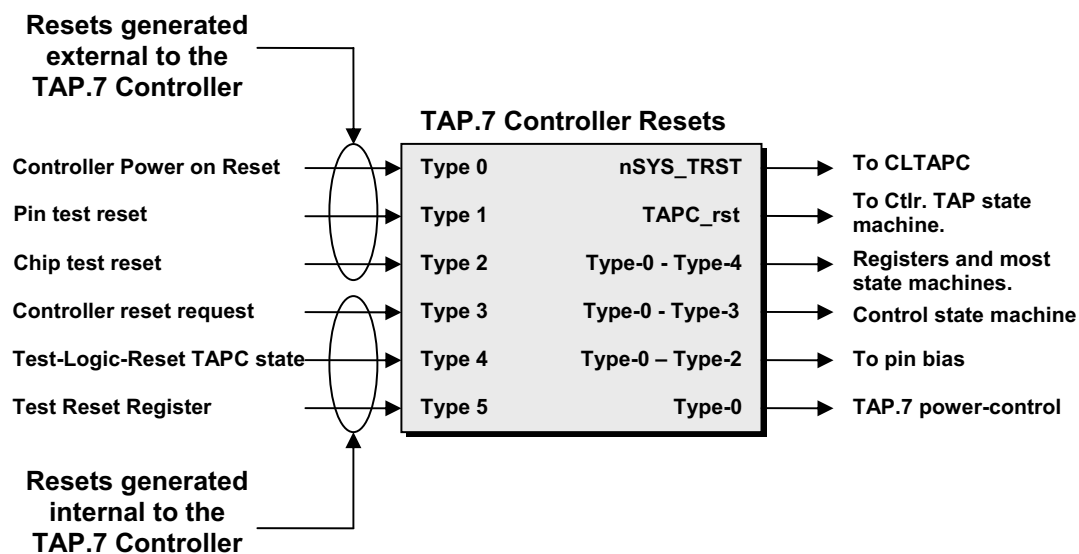


Figure 1605. Conceptual view of the EPU reset logic and state machines

67.5.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

67.5.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

67.5.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

67.5.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

67.5.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

67.5.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

67.5.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

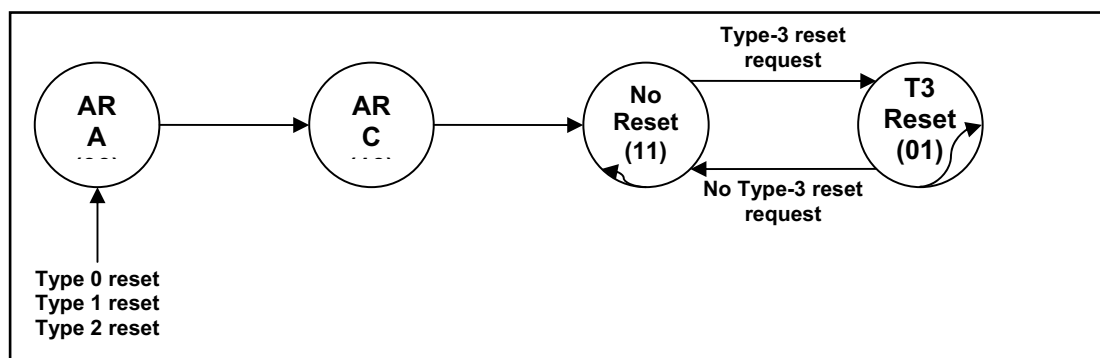


Figure 1606. Reset state machine

Table 1503. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released

Table 1503. Reset state descriptions

Reset state		Description
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in [Table 1504](#).

Table 1504. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	–	–	–	–	–	–
Type-4	–	Yes	Yes	–	–	–	–
Type-3	Yes	Yes	Yes	Yes	Yes	–	–
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

67.5.3 Start-up Options

67.5.3.1 Overview

The TAP.7 start-up options are shown in [Table 1505](#). The only startup option currently supported by the CJTAG module is 1149.1 Compliant.

Table 1505. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to <i>Run-Test/Idle</i>

67.5.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine.
- The TDIC and TDOC pins are present and provide the TDI and TDO function.
- The CLTAPC is coupled at start-up.

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 1506. Reset values for TAP.7 start-up options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMS pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

67.5.4 RSU operation

The RSU uses TCKC and TMS with the control protocol for configuration control.

67.5.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP.7 controller is Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in [Figure 1607](#).

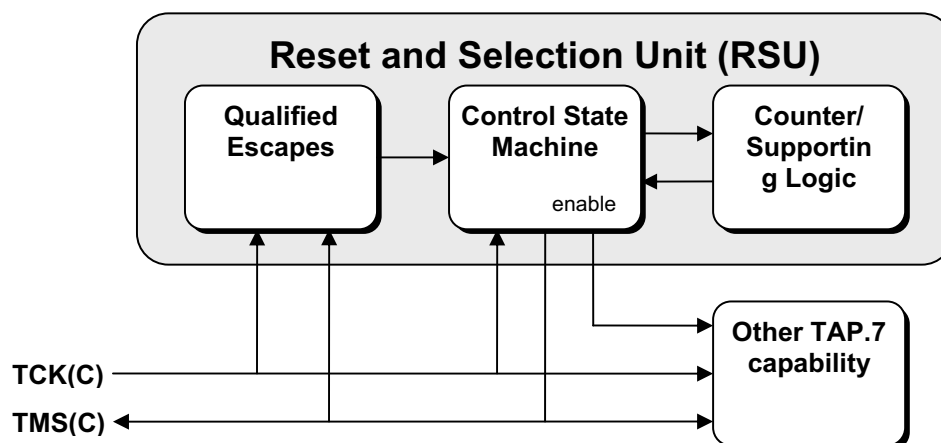


Figure 1607. Conceptual view of technology selection mechanism

67.5.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

67.5.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges)– Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges)– Resets all technologies (generates a Type-3 TAP.7 TAPC reset)

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

67.5.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

67.5.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occur with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

67.5.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not

contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

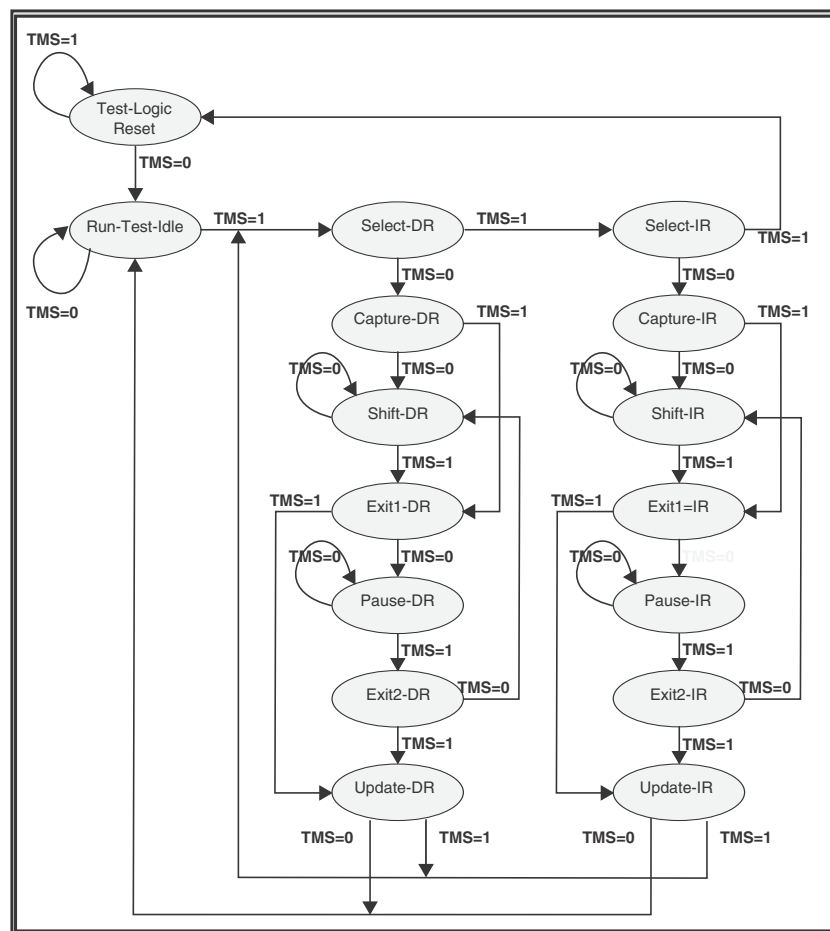


Figure 1608. Falling edge TAP controller state machine

67.6 EPU (Extended Protocol Unit) Operation

67.6.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

67.6.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

67.6.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

67.6.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

67.6.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Accesses to optional TAP.7 scan paths
6. Force offline operation (optional)
7. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control level.

67.6.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

67.6.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

67.6.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called “**TAP.7 ZBSs**”. ZBSs that are intended for other uses are henceforth called “**Other ZBSs**”. Two TAP.7 registers enable the STL’s use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP.7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

67.6.2 EPU Registers

67.6.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose. These commands store a single global register. Global registers manage TAP.7 TAPC

functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

67.6.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one at a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers. There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back..

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- Read-only – Configuration and register read back.

Table 1507. TAP.7 controller register list (managed with commands)

Register type	Global/ local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
Options	Local	1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Select	Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge
Read-only	Local	32	RDBACK0	Read-back 0
		32	CNFG0	Configuration Register 0
M = Mandatory register O = Optional register				

67.6.2.3 Registers

67.6.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

67.6.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the "System Path" (STL) is used.

- **1** – Suspend use of control levels. Enables "System Path."
- **0** – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

67.6.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

- **1** – Inhibit ZBS detection.
- **0** – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

67.6.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 1508. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	JScan0: – 1149.1 Compliant Operation. – Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	JScan1: – 1149.1 Compatible Operation – Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	JScan2: – 1149.1 Compatible Operation. – Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register
T3–T4	00011	JScan3: – Non-compatible 1149.1 Operation – Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. – Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. – Enables the use of SSDs

Table 1508. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
T4	01000	OScan0 – Data rate dependent components – Single TAP.7 communication – Debug and Test applications
	01001	OScan1 – TAP.1 components – Single/Multi-TAP.7 communication – Debug and Test applications
	01010	OScan2 – TAP.1 components – No TDI or TDO in non-shift states – Debug applications
	01011	OScan3 – TAP.1 components – No TDI in non-shift states – No TDO in any state
T4	01100	OScan4 – Data rate dependent components – Single TAP.7 communication – Debug and Test applications
	01101	OScan5 – TAP.1 components – Single/Multi-TAP.7 communication – Debug and Test applications
	01110	OScan6 – TAP.1 components – No TDI or TDO in non-shift states – Debug applications
	01111	OScan7 – TAP.1 components – No TDI in non-shift states – No TDO in any state
	10000	MScan – Data rate dependent components – Multi TAP.7 communication – Directed CID assignment – Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	Reserved

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

67.6.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

- **0** – Scan selection directives are not enabled.
- **1** – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

67.6.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance the external tool TAPC state, receive TMS and TDI information, and begin generation of a new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0

Bits 18-17 of the “Global Register.”

67.6.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

67.6.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

- **0** – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.
- **1** – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1

value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0.
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

67.6.2.3.9 Auxiliary Pin Function Control (APFC)

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

- **0** – TDI and TDO have JTAG function.
- **1** – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

67.6.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

67.6.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

67.6.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0

67.6.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as “0”.

Table 1509. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

67.6.2.3.14 Configuration Register 0 (CNFG0)**Table 1510. Configuration Register 0 Format**

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class
				0000 – TAP.1 or TAP.7 Class T0
				0001 – Class T1
				0010 – Class T2
				0011 – Class T3
				0100 – Class T4
				0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision.
				0000 – Reserved
				0001 – IEEE Std 1149.7-2008
				0010 – 1111 - Reserved

Table 1510. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format
				xxx1 – CNFG0[31:12] are implemented
				xx1x – CNFG1 implemented
				x1xx – CNFG2 implemented
				1xxx – CNFG3 implemented
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 – Yes

Table 1510. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 - Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported
Reserved	31:29	3	Reserved	Reserved, read as zero

67.6.2.3.15 Register reset values

Table 1511. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DSG M	DSG M	DSG M	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

67.6.3 EPU Commands

There are two command types:

- Two Part Commands
- Three Part Commands

Any number of two-part and three part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

67.6.3.1 Two Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two). The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

67.6.3.2 Three Part Commands

Some commands are used to send and/or receive data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

67.6.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of *Shift-DR* states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in [Figure 1609](#).

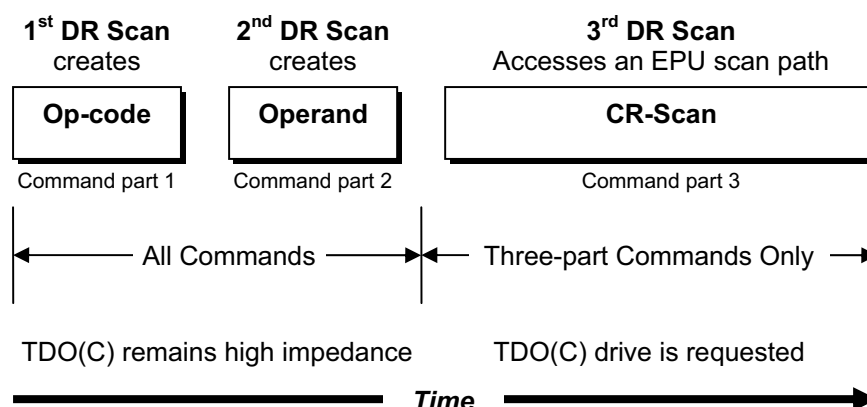


Figure 1609. DR_Scan sequence used for command creation

The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a **Control-Register Scan** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two part commands, but only three three-part commands, each having a special purpose.

67.6.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in [Table 1512](#).

Table 1512. TAP.7 controller command list

Command class	Op-code count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	-	M	M	M
	0x04	STDCST	Store Data Channel State	-	-	-	-
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	-	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate (1)	0x0A	CIDA	Allocate Controller ID	-	-	M	M
	0x0B	CIDD	De-allocate Controller ID	-	-	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	-	-	-	-

1. Enumerate commands are not currently supported by the CJTAG module.

67.6.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing a external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

67.6.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller's CID is valid, and are treated as no operations otherwise.

67.6.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

67.6.3.5 Command Definitions

67.6.3.5.1 Store Commands

67.6.3.5.1.1 Store Miscellaneous Control Command (STMC)

Table 1513. Store Miscellaneous Control

STMC		Store Miscellaneous Control				
Opcode	Operand	bbb		x	y	Description
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	-	SGC = y
				1	-	CGM = y
		010	RdyCtl	-	-	RDYC = xy
		011	DlyCtl	-	-	DLYC = xy
		100	ScnFunc	0	-	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	-	-	Reserved
		110				
		111				

67.6.3.5.1.2 Store Conditional 1-bit (STC1)

Table 1514. Store Conditional 1 bit (STC1)

STC1		Store Conditional 1 bit			
Opcode	Operand	c	bbb	CAPE	Description
00001	c bbb v	0	000	-	SREDGE = v
			001	-	Reserved
			010	-	TRESET = v
			011 – 111	-	Reserved
		1	--	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

67.6.3.5.1.3 Store Conditional 2-bit (STC2)

Table 1515. Store Conditional 2 bit (STC2)

STC2		Store Conditional 2 bit			
Opcode	Operand	c	bb	CAPE	Description
00010	c bb vv	0	00	-	Reserved
			01	-	STCKDC = vv
			10	-	APFC = vv
			11	-	Reserved
		1	—	0	No change
			00	1	Reserved
			01	1	STCKDC = vv
			10	1	APFC = vv
			11	1	Reserved

67.6.3.5.1.4 Store Scan Format (STFMT)

Table 1516. Store Scan Format (STFMT)

STFMT		Store Scan Format
Opcode	Operand	
00011	nnnnn	SCNFMT = nnnnn

67.6.3.5.2 Select Commands

67.6.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 1517. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	Operand	m	Description
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

67.6.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controller's CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 1518. Make Scan Group Candidate (MSC)

MCM		Conditional Group Member	
Opcode	Operand	m	Description
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

67.6.3.5.3 Scan Commands

67.6.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 1519. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	Operand	CR-Scan	yyyyy	Description	R/W
01000	yyyyy	-	0	Scan Group Candidate (SGC)	Write only
		-	1	Conditional Group Member (CGM)	Write only
		-	2	CNFG[0]	Read only
		-	3	CNFG[1] (not implemented)	Read only
		-	4	CNFG[2] (not implemented)	Read only
		-	5	CNFG[3] (not implemented)	Read only
		-	6	Functional reset requested (FRESETR) (not implemented)	Read only
		-	7	Series topology detect	Write only
		-	15 – 8	Reserved	Read only

67.6.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

Table 1520. Scan String (SCNS)

SCNS			Scan String		
Opcode	Operand	CR-Scan	yyyyy	Description	R/W
01001	yyyyy	-	0	RDBACK[0] Register [31:0]	Read only
		-	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		-	2	CNFG[0] Register [31:0]	Read only
		-	3	CNFG[1] Register [31:0] (not implemented)	Read only
		-	4	CNFG[2] Register [31:0] (not implemented)	Read only
		-	5	CNFG[3] Register [31:0] (not implemented)	Read only
		-	6 - 7	Reserved (read logic 0)	Read only
		-	31 - 8	Private – available for customization	

67.6.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in [Figure 1610](#).

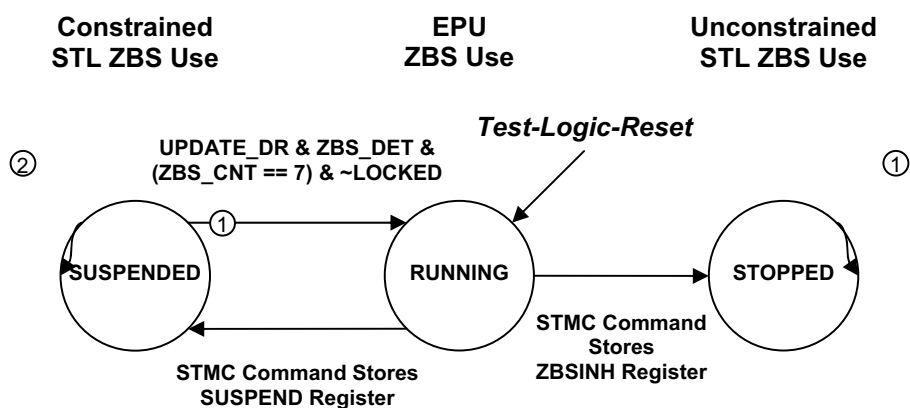


Figure 1610. EPU operating states

The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in [Table 1521](#). The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 1521. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating state	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

67.6.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path
- **Bit** – Supports set and test bit operations (a 1-bit shift register)
- **String** – Support 32-bit reads or 32-bit writes
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

67.7 APU (Advanced Protocol Unit)

67.7.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMS signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (i.e., ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine state progression. The EPU is unaware of the APU's existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Three groups of scan formats (MScan, OScan0-7, and SScan0-3) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan and SScan Scan Formats are broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called "offline". A TAP.7 controller that is not offline is henceforth called "online". While offline the TAP.7 controller awaits the detection of a special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

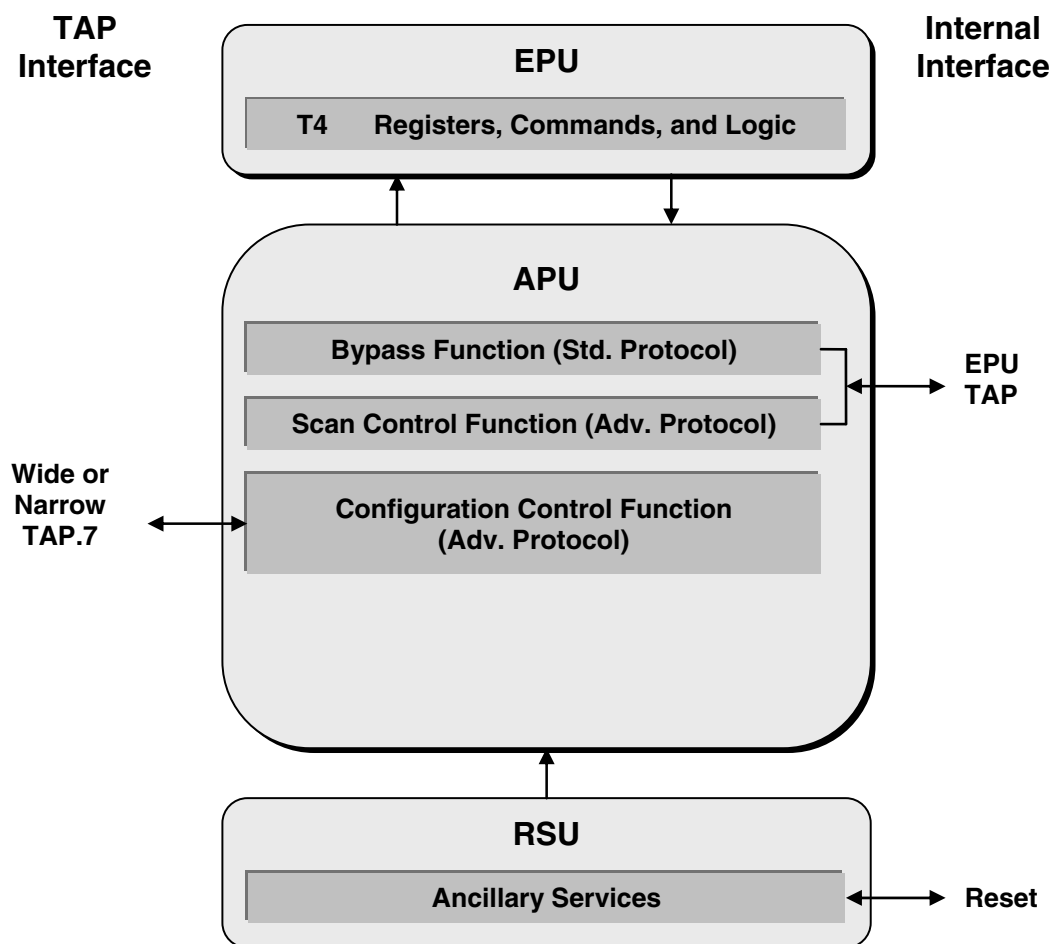


Figure 1611. Conceptual block diagram of the APU functions

The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure (ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

67.7.2 Operation

The conceptual view of the APU operation is shown in [Figure 1612](#).

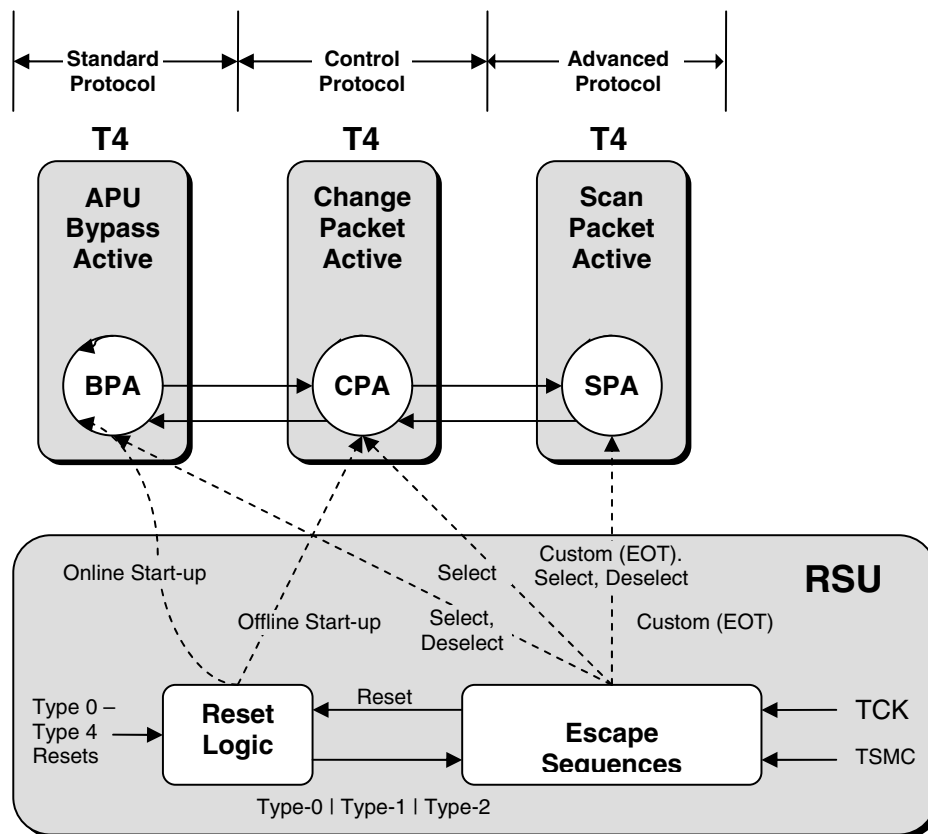


Figure 1612. Conceptual view of APU operation

The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMS pin.

APU operating states:

- **BPA** – ByPass Active
- **CPA** – Change Packet Active
- **SPA** – Scan Packet Active

67.7.2.1 BPA

The *BPA* state allocates the use of the TMS pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

67.7.2.2 CPA

The CPA state allocates the use of the TMS pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

67.7.2.3 SPA

The SPA state allocates the use of the TMS pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

67.7.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMS pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

67.7.4 Signal behaviors

The TCKC signal:

- The test clock

The TMS signal:

- An input when the Standard Protocol is used
- Bi-directional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
 - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
 - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

67.7.5 APU Functions

67.7.5.1 Bypass Active (BPA) Function

When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (e.g., a narrow TAP.7

or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

67.7.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in . There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

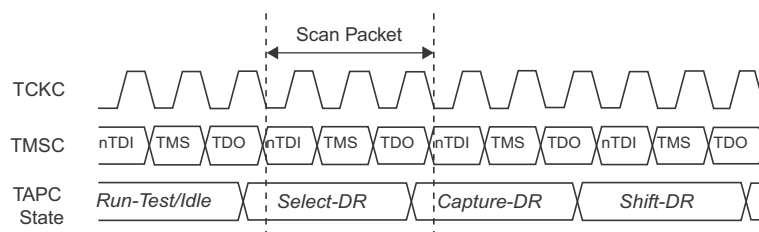


Figure 1613. Scan Packet sequence

A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

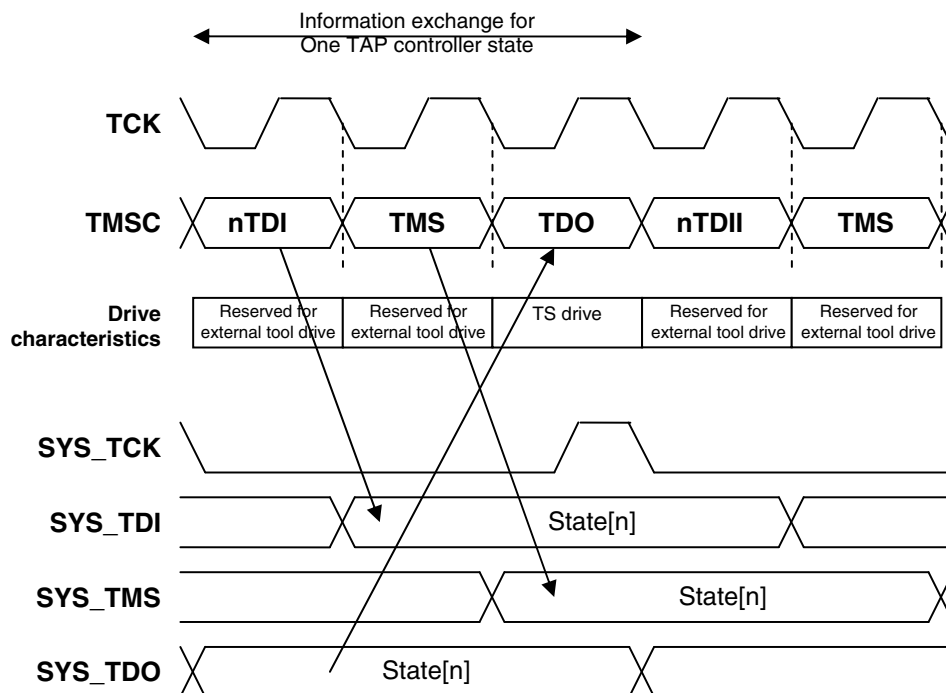


Figure 1614. Serialization and De-serialization of Scan Packet Information

In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats.)

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

67.7.5.3 Control Process Active (CPA) Function

While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

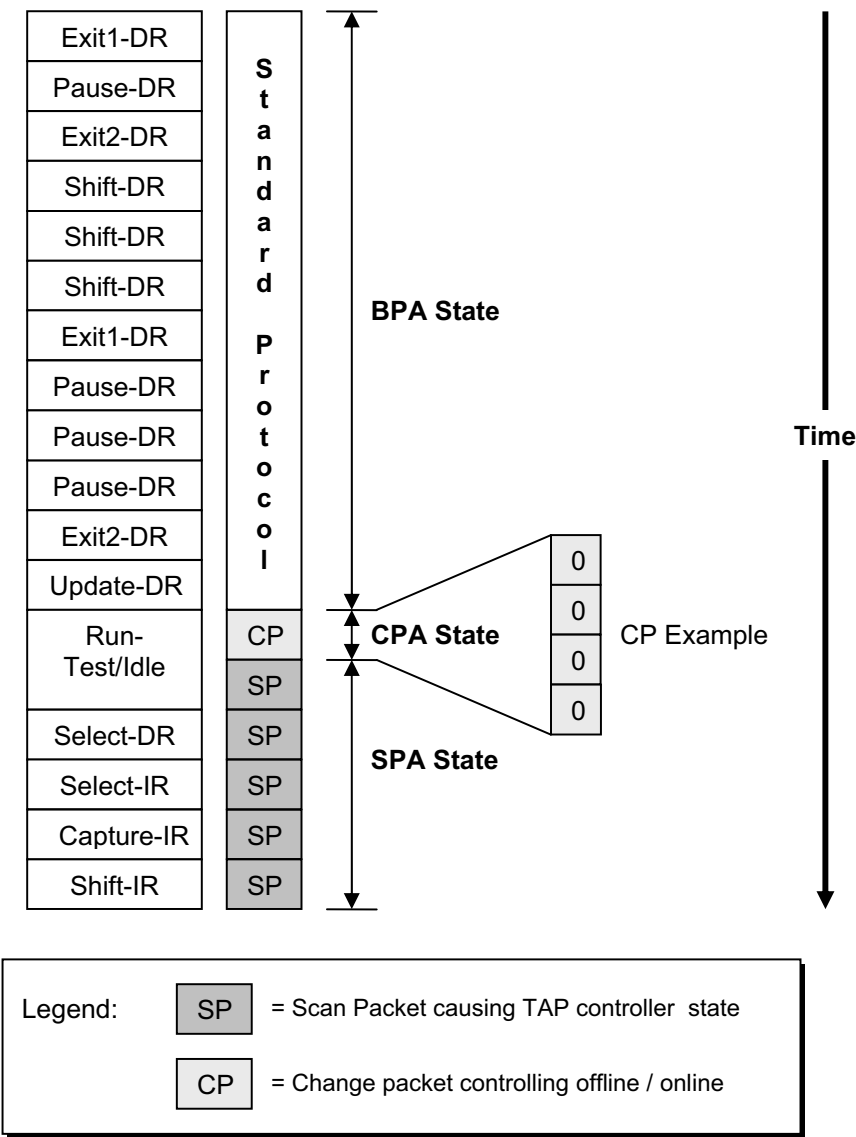


Figure 1615. CPA State Entry from BPA State with Continued Online Operation

An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

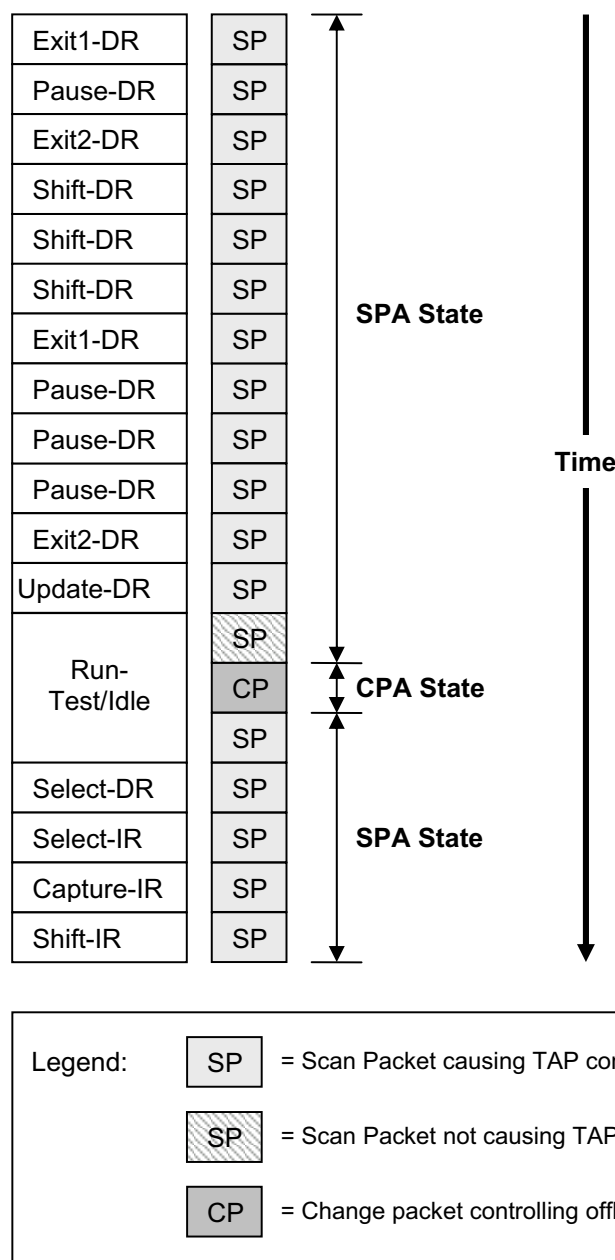


Figure 1616. CPA State Entry from the SPA State with Continued Online Operation

When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

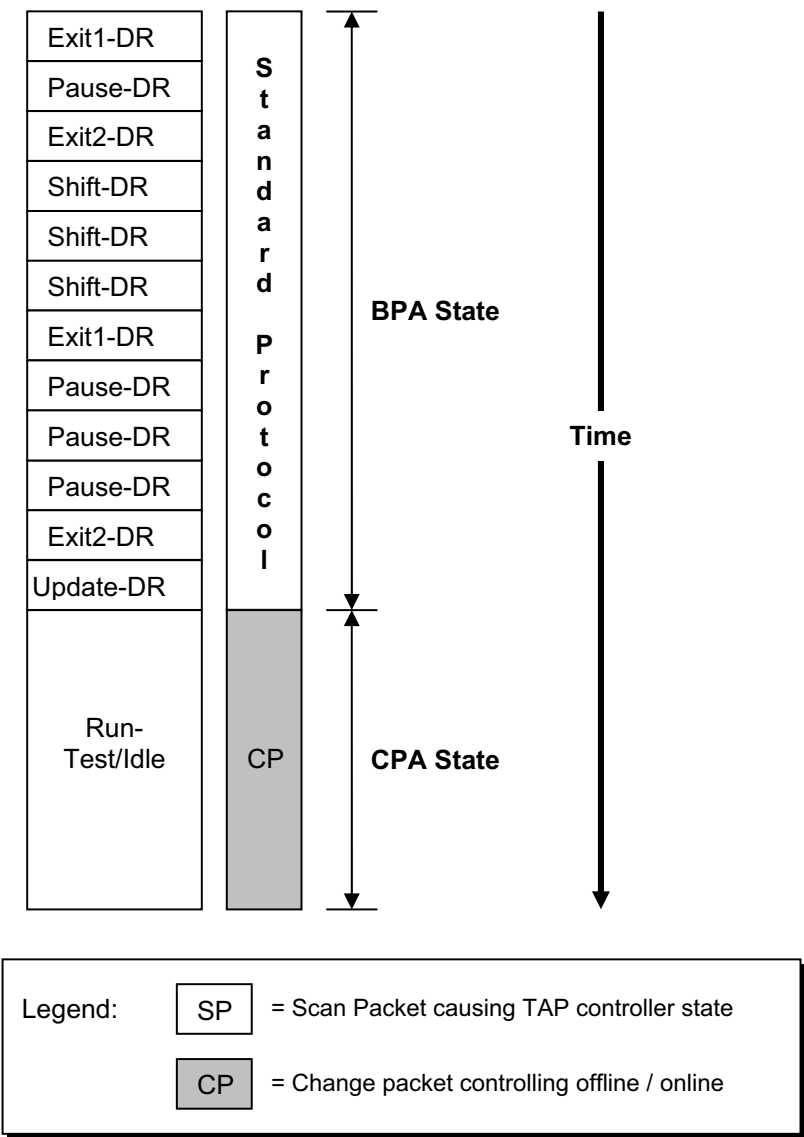


Figure 1617. CPA State Entry from the BPA State with Offline Operation

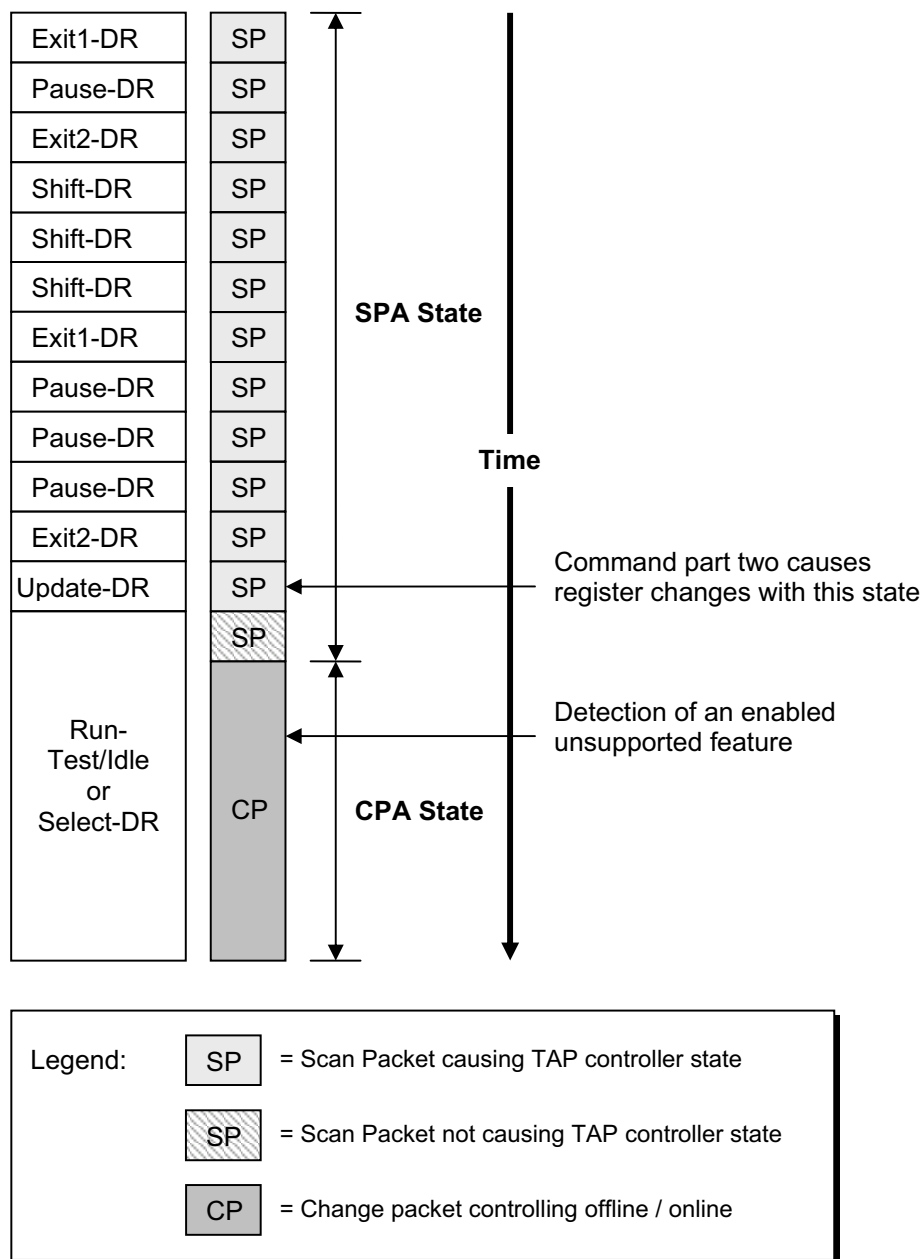


Figure 1618. CPA State Entry from the SPA State with Offline Operation

A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

67.7.6 Configuration Change Packets (CP)

A Change Packet (CP) is comprised of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.
- **Body** – Two or more bits representing one or more CP directives
 - **00** – CP_END
 - **01** – CP_NOP (extends the body by 1-bit)
 - **10** – CP_NOP (extends the body by 1-bit)
 - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the postamble bit).
- **Postamble** – A 1-bit value that is a don't care. This bit facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Postamble. This can be as simple as four logic zero values.

67.7.7 Scan Packet

A Scan Packet defines the period of TMS pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof)
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

67.7.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

67.7.8.1 Payload Element Content

67.7.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain an input bit frame, an output bit frame, or both. When an input bit frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

67.7.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in [Table 1522](#).

Table 1522. Factors determining the SP payload content

Scan format	Payload content determined by the :	
	TAPC state	Header
MScan	—	—
OScan	Yes	—

67.7.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

67.7.8.3 Scan Formats

67.7.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use

cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in [Table 1523](#) and [Table 1525](#), respectively. The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

67.7.8.3.2 MScan scan formats

The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

Table 1523. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			
	Multi-chip debug communication			
	Controller ID Allocation			

Table 1524. MScan Scan Packet content

MScan	Non-shift TAP controller states						Shift TAP controller states					
	nTDI	TMS	PC0	RDY	PC1	TDO	nTDI	TMS	PC0	RDY	PC1	TDO

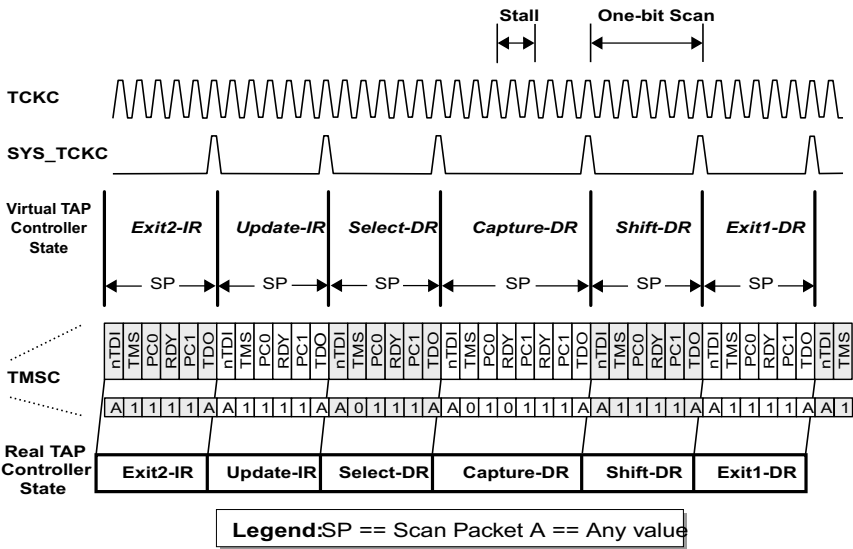


Figure 1619. MScan Transaction

67.7.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

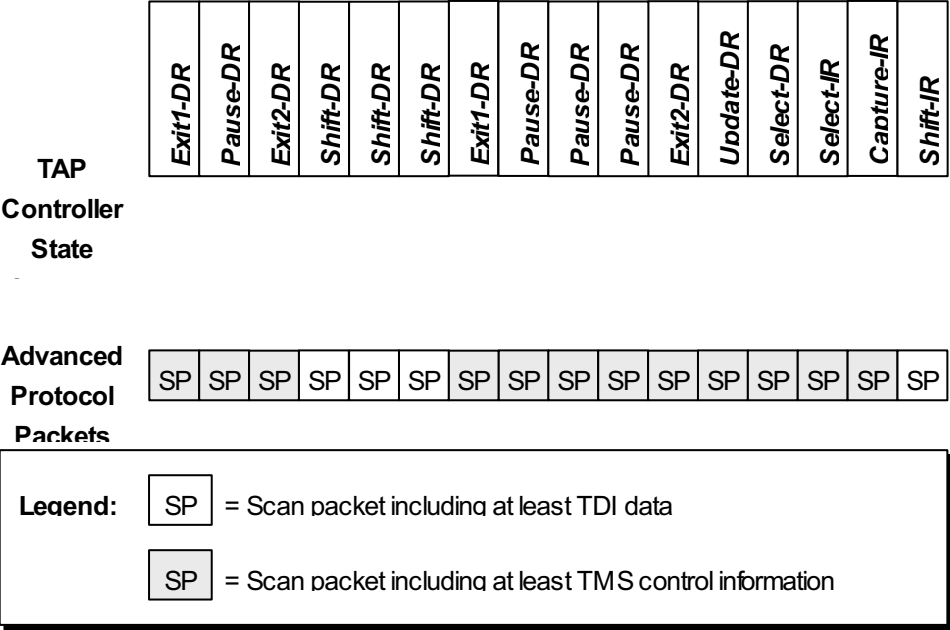


Figure 1620. OScan Scan Packet content/TAPC state relationships

OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan 4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, an EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods for this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in [Table 1525](#).

Table 1525. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best		External tool or TS
OScan4	Stalls	↓	Best	
OScan5	Test		↑	
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 1526. OScan Scan Format “Payload”

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	–	TDO		nTDI	TMS	–	TDO
OScan2	–	TMS	–	–		nTDI	TMS	–	TDO
OScan3	–	TMS	–	–		nTDI	TMS	–	–
OScan4	nTDI	TMS	RDY	TDO		nTDI	–	RDY	TDO
OScan5	nTDI	TMS	–	TDO		nTDI	–	–	TDO
OScan6	–	TMS	–	–	Note 1	nTDI	TMS	–	TDO
					Note 2	nTDI	–	–	TDO
OScan7	–	TMS	–	–		nTDI	–	–	–

Note 1: Following Capture-xR or Exit2-xR Scan Packet
Note 2: Following Shift-xR Scan Packet

68 JTAG Data Communication (JDC)

68.1 Introduction

The JTAG Data Communication (JDC) module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory mapped register space and an external tool that accesses the JTAG port.

68.2 Overview

The JDC module consists of IPS accessible registers, JTAG accessible registers, and associated logic to coordinate movement of data from one register domain to another.

The JDC implements the following IPS data registers, occupying separate memory space:

- 32-bit memory mapped register that can be read or written via IPS (JOUT_IPS), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 32-bit memory mapped register that can only be read via IPS (JIN_IPS), and whose contents are loaded from a JTAG register (JIN).

In addition to the data registers themselves, logic is implemented to indicate when new data has been shifted in via the JTAG port and is ready to be read from the JIN_IPS register, and when new data has been written to the JOUT_IPS register and is ready to be read via the JTAG port. The state of these flags is captured into a status register (MSR), and also provided to a JTAG register (JOUT) for tool visibility.

The JTAG accessible registers are provided through an instance of the existing JTAC module.

There is a single bus interface to port register data out to the JTAC, and a single bus interface to port data in from the JTAC. The architecture is shown in [Figure 1621](#).

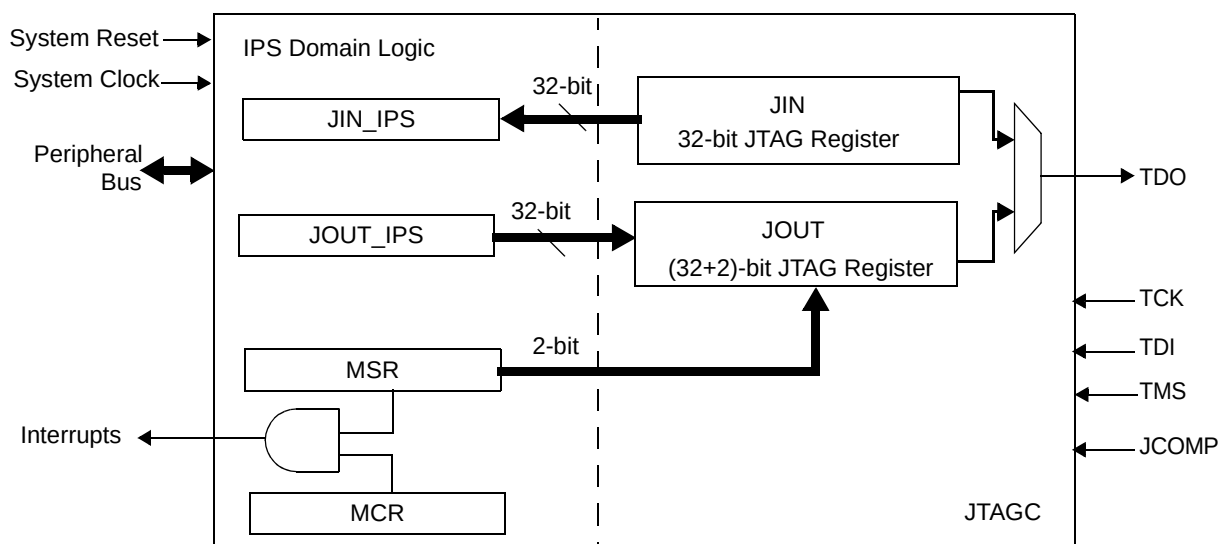


Figure 1621. JDC block diagram

68.3 Register definition

[Table 1527](#) shows the JDC module registers. Four 32-bit registers are implemented. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

Table 1527. JDC memory map

Address offset	Register	Access	Reset value	Location
0x0	Module Configuration Register (MCR)	R/W	0x0000_0000	on page 2971
0x4	Module Status Register (MSR)	R/W	0x0000_0000	on page 2971
0x8	JTAG Output Data Register (JOUT_IPS)	R/W	0x0000_0000	on page 2972
0xC	JTAG Input Data Register (JIN_IPS)	Read Only	0x0000_0000	on page 2973

68.3.1 Register descriptions

68.3.1.1 Module configuration register (MCR)

[Figure 1622](#) shows the MCR. The MCR is used to enable the interrupt outputs of the JDC.

Figure 1622. Module Configuration Register (MCR)

Offset 0x00																Access: User read/write																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	JIN_IEN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	JOUT_IEN
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The MCR is described in [Table 1528](#).

Table 1528. MCR register field descriptions

Field	Description
JIN_IEN	JIN Interrupt Enable. 1 Setting of MSR[JIN_INT] bit asserts the JIN interrupt 0 Setting of MSR[JIN_INT] bit does not assert the JIN interrupt
JOUT_IEN	JOUT Interrupt Enable. 1 Setting of MSR[JOUT_INT] bit asserts the JOUT interrupt 0 Setting of MSR[JOUT_INT] bit does not assert the JOUT interrupt

68.3.1.2 Module status register (MSR)

[Figure 1623](#) shows the format of the MSR. The MSR holds the JTAG register status and interrupt bits.

Figure 1623. Module Status Register (MSR)

Offset 0x04																Access: User read/write																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	JIN_RDY	0	JIN_INT	0	0	0	0	0	0	0	0	0	0	0	0	0	JOUT_RDY	0	JOUT_INT
W																w1c																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The MSR is described in [Table 1529](#).

Table 1529. MSR register field descriptions

Field	Description
JIN_RDY	JIN Ready (read only). 1Set when new data is written to the JIN_IPS register 0Cleared upon software read of JIN_IPS contents via IPS
JIN_INT	JIN Interrupt. 1Set when new data is written to the JIN_IPS register 0Cleared by writing logic 1
JOUT_RDY	JOUT Ready (read only). 1Set when new data is written to the JOUT_IPS register 0Cleared upon tool read of JOUT register via JTAG port
JOUT_INT	JOUT Interrupt. 1Set when JOUT_RDY bit is cleared by tool reading JOUT register 0Cleared by writing logic 1

68.3.1.3 JTAG output IPS data register (JOUT_IPS)

The JOUT_IPS register holds data written via IPS. The JOUT_IPS contents are ported out and captured into the JOUT register to be read via the JTAG port. [Figure 1624](#) shows the format of the JOUT_IPS register.

Figure 1624. JOUT_IPS register

Offset 0x08																Access: User read/write																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	JOUT_IPS Data																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The JOUT_IPS register is described in [Table 1530](#).

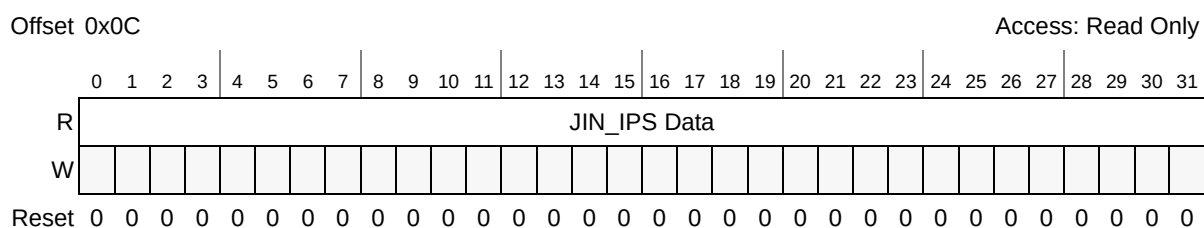
Table 1530. JOUT_IPS register field descriptions

Field	Description
JOUT_IPS Data	Contains JOUT_IPS data.

68.3.1.4 JTAG input IPS data register (JIN_IPS)

Data written to the JTAG input data register (JIN) via the JTAG port is held in the JIN_IPS register, where it can be read via IPS. [Figure 1625](#) shows the format of the JIN_IPS register.

Figure 1625. JIN_IPS register



The JIN_IPS register is described in [Table 1531](#).

Table 1531. JIN_IPS register field descriptions

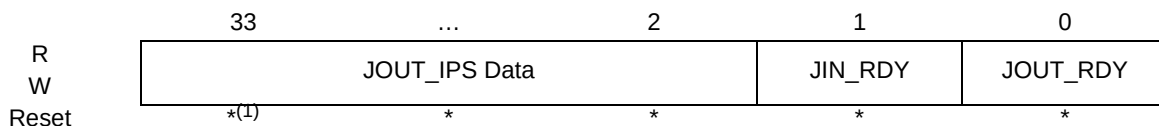
Field	Description
JIN_IPS Data	Contains JIN_IPS data.

68.3.2 Non-Memory mapped register definition

The JDC also implements two JTAG accessible registers that are not memory mapped. One JTAG register is used to shift in data to be placed in the JIN_IPS register. The other JTAG register captures data from the JOUT_IPS register and ready status from the MSR to be shifted out via the JTAG port.

68.3.2.1 JTAG output data register (JOUT)

The JOUT register captures data from the JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JIN_RDY and JOUT_RDY status bits. [Figure 1626](#) shows the format of the JOUT register.



1. The reset value of the JOUT register is 34'b0.

Figure 1626. JOUT JTAG register

The JOUT register is described in [Table 1532](#).

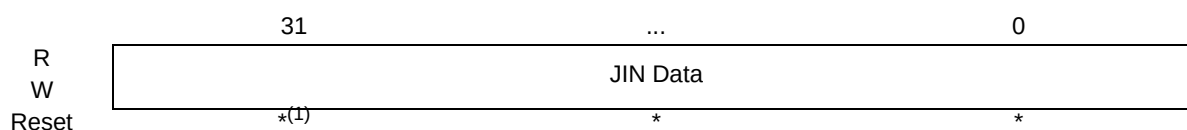
Table 1532. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from MSR
JOUT_RDY	State of JOUT_RDY bit from MSR

68.3.2.2 JTAG input data register (JIN)

Data is written to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JIN data is later captured in the JIN_IPS register to be read via IPS.

[Figure 1627](#) shows the format of the JIN register.



1. The reset value of the JIN register is 32'b0.

Figure 1627. JIN JTAG Register

The JIN register is described in [Table 1533](#).

Table 1533. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

68.4 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data into a memory mapped register that can be accessed via IPS. It also provides the ability to capture data written to a memory mapped register into a JTAG shift register for output via the JTAG port. An overview of the module functionality is described below.

- A software write to the JOUT_IPS register sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port. The state of the JOUT_RDY bit is reflected in the MSR and also ported out to the JOUT register. A JTAG read of the JOUT register via execution of the JOUT_READ instruction with a JOUT_RDY bit whose value is logic 1 indicates the register contains new data. The JOUT_RDY flag bit is cleared upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.
- A JTAG write to the JIN register via execution of the JIN_READ JTAG instruction updates the contents of the JIN_IPS register upon exit of the Update-DR state. An update of the JIN_IPS register sets the JIN_RDY flag bit, indicating new data is available to be read via IPS. The state of the JIN_RDY bit is reflected in the MSR register and also ported out to the JOUT register. The JIN_RDY flag bit is cleared upon software read of the JIN_IPS register. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates new data can be written to the JIN register.

68.4.1 IPS register access

IPS access is available to the registers described in [Section 68.3.1, Register descriptions](#).

68.4.2 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

68.4.2.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in [Table 1534](#). This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 1534. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JIN_IPS upon exit of Update-DR state while JIN_WRITE is active.
BYPASS	11111	Selects bypass register for data operations
Reserved	All other opcodes	Decoded to select bypass register

68.5 Use case example

A summary of how the JDC can be used to interact with the Hardware Security Module (HSM) to perform challenge/response password authorization is as follows:

1. After exit of reset, all registers are at their reset values. JIN_RDY bit value of 0 indicates the tool may write data to the JIN register. JOUT_RDY bit value of 0 indicates that software may write data to the JOUT_IPS register.
2. The MCR register is programmed to enable JOUT and JIN interrupts, or not.
3. Once the tool is ready to start the authorization process, it writes a value to the JIN register via execution of the JTAGC JIN_WRITE instruction.
4. Software monitors the state of the JIN_RDY bit or enables the JIN interrupt and uses the interrupt assertion as a trigger to start the authorization process.
5. Once the authorization process is started, software writes the first JOUT value to the JOUT_IPS register. This sets the JOUT_RDY bit value to 1.
6. Following the initial tool write to the JIN register, the tool polls the JOUT register to determine when software has provided a challenge word. Upon reading the JOUT register with JOUT_RDY bit set, the tool records the data value to be used to generate the appropriate response. The read of the JOUT register clears the JOUT_RDY bit.

The clearing of the JOUT_RDY bit also sets the JOUT_INT bit and asserts the JOUT interrupt if the MCR[JOUT_IEN] is set.

7. With the JOUT_RDY bit cleared, software may now provide the next challenge word.
8. Once the tool has generated a response word, it uses the value of the JIN_RDY bit from the JOUT register read to determine when it can perform a write to the JIN register to provide the response.
9. The HSM can provide additional challenge words and read back the corresponding response words as needed, repeating the process.

Simply providing a password without using the challenge/response protocol can be done in a similar way. The tool provides 32-bits of the password at a time with each write to the JIN register, and monitors the JIN_RDY bit value of the JOUT register to determine when the next JIN register value can be written. There is no need for software to write the JOUT_IPS register in this case.

69 Sequence Processing Unit (SPU)

69.1 Introduction

The Sequence Processing Unit (SPU) provides an on-device trigger functions similar to those found on a logic analyzer. In a complex SoC, performance monitor functions are available at two levels: system level and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System level performance monitor functions are integrated into the SPU complex trigger logic, and thus allow counting and timing of any debug trigger combinations supported by the SPU.

Various clients such as the cores, GTM, or crossbar switch, generate watchpoints and triggers when operating in their debug mode. The SPU collects these triggers (such as interrupt occurrence, address watchpoint, and so on) and uses them as conditions to sequence through states, with resultant actions (such as start/stop trace, start/stop counter, capture timebase, and so on). The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, complex logic-analyzer-like behavior can be achieved, providing vital real-time visibility and debug ability of the activities of the system.

69.1.1 SPU block diagram

Figure 1628 shows a top-level block diagram of the SPU.

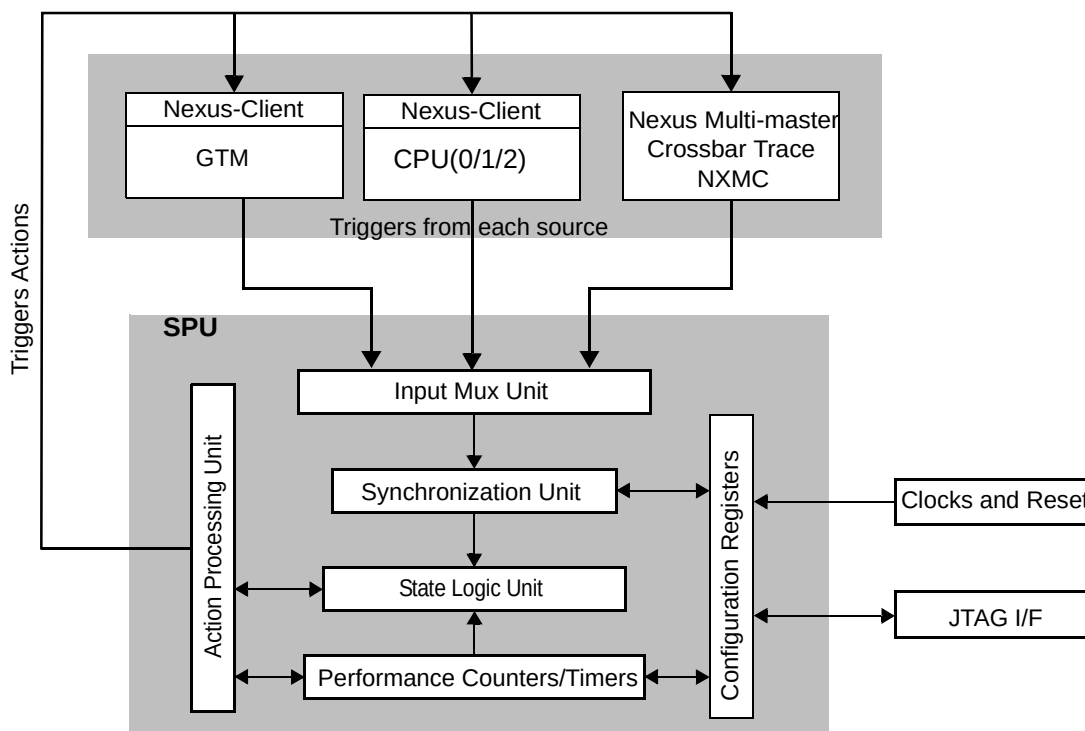


Figure 1628. SPU block diagram

69.1.2 Overview

The SPU is able to generate complex debug events, based upon input events from sources (CPUs, GTM, and system bus clients) throughout the SoC. The SPU has the ability to create a state machine to trigger a debug action. Single or multiple actions can be triggered by a state machine, which results in debug events being created. There are a selection of timers and counters available to assist comparison events.

Complex trigger support is implemented in the State Logic Unit (SLU). There are numerous inputs coming from different clients. Users can choose 63 inputs from a set of 209 inputs at the level one Mux. There are actually 64 inputs to the Muxes but one of the inputs is permanently connected to a high level, so users can only choose 63 of the inputs. Out of these 63 inputs, users can select 16 inputs for each SLU. This is shown in [Figure 1667](#).

One SLU consists of four input AND gates, the output of which are ORed together. One of the inputs of each AND gate has the option to invert it. Also, the output of each AND gate can be optionally inverted. Finally, the output of each SLU has the option of inversion. The block diagram for one SLU state is shown in [Figure 1668](#). The SPU includes eight SLUs states.

Users can utilize these states to form a configurable state machine. This allows users to join states together with the if-then-else operations to create a sequence. Each state in a sequence has the ability to move to another state based on the true condition from the state logic, and the ability to route to another state based on the a false condition from the state logic. This is shown in [Figure 1669](#) and [Figure 1670](#). At a given time, a maximum of four sequences are allowed. Each sequence can have one state or maximum of eight states. Each state can only be used in one unique sequence. The number of states in all the sequences cannot exceed eight. Each state has the ability to trigger one to four actions based on a true/false condition in the active sequence.

An example of typical complex trigger for the SPU would be: if FunctionA is entered, monitor CPU0 writes to VariableB. If CPU0 writes a value to VariableB which is 0x100, then enable trace for the GTM and halt CPU 0.

The SPU consists of the following sub-blocks:

- Input Mux Unit—receives the numerous triggers from various clients and reduces them down into a user selected set. The muxing unit selects sixteen inputs for each sequencing SLU from a set of 63 static inputs. This selection is done using two levels of muxing. At the first level of muxing, users select 63 out of 209 inputs. At the second level of muxing, out of these 63 inputs, one input can be selected for each of the four input AND gates. There are 128 64×1 Muxes to cater the needs of 128 inputs of all the eight states. This input mux selection logic is shown in [Figure 1667](#).
- State Logic Units (SLU)—responsible for performing operations on the selected input events to produce desired events. Each SLU has a control register that manages how the various inputs are combined to produce a desired event. Only one input of each four input AND gate can be independently inverted. The combined event can also be optionally inverted. These optional inversions coupled with DeMorgan's law allows for generalized signal combinations. This is shown in [Figure 1668](#).
- Action Processing Unit—takes an action request from a SLU state and converts this into one or multiple actions. The action unit can be programmed to trigger various actions in response to each processed event from the SLU unit. The user can define the actions associated with each state from each SLU.
- Counters/Timers—The SPU implements 16 32-bit timer/counter functionality. Timer or counter selection is done via a configurable register in the SPU. Each counter/timer

has its own control register. A 32-bit comparator value can optionally be compared against the timer/counter. These registers are accessible via JTAG.

69.1.3 Features

The features of the SPU are as follows:

- clients:
 - 3 CPUs
 - 1 GTM
 - 2 System Bus clients(NXMC)
- Input watchpoints/triggers selection
- Input debug triggers for instruction execution selection
- Input interrupt selection
- Sequences and states criteria
 - Up to 4 user programmable number of active/simultaneous sequences
 - Up to 8 user programmable states that can be split between all four sequences
 - Each state can only be used in one unique sequence
 - A sequence can consist of a single state or any number of states up to 8
- State logic
 - Each state logic consist of sum of products (SOP)
 - SOP consist of four 4-input AND gates, and one 4-input OR gate
 - Optional inversion on one input of each AND gate of each state logic
 - Optional inversion on output of each AND gate.
 - Optional inversion on output of OR gate of each state
 - States are joined together with if-then-else statement to create a sequence
- SLU events
 - Up to 8 user programmable derived events/actions
 - Any state can optionally trigger of 1 to 4 actions based on a true condition
 - Any state can optionally trigger of 1 to 4 actions based on a false condition
 - User selection of trigger actions generated in response to each programmed trigger

- SPU counters/timers
 - 16 32-bit counters/timers
 - Timer/counter configurability selection via a register
 - Start/increment when used as timer/counter respectively
 - Reset when used as timer/counter respectively
 - Counter value compare match
 - Counter overflow event status
 - Generation of a greater than condition against a 32-bit comparator value (no support for range condition from single counter)
 - Range support using two counters
 - Use of counter/timer registers to generate counted or timed triggers
 - Use of counter/timer registers to capture system performance statistics
 - Each timer/counter value has the ability to be inserted into the Nexus trace stream upon an SPU action
 - Access to timer/counter values via JTAG
- Prescaler for timer mode only
 - Configurable prescaler of /1, /4, /16 or /32 of the clock for timer operation inside the SPU
- Asynchronous interfaces
 - SPU to/from CPU (fastest clock)
 - SPU to/from NXMC
 - SPU to/from GTM
 - SPU to/from NAR
 - SPU to/from DCI
 - SPU to/from JTAGM
- Interfaces
 - CPU
 - GTM
 - System bus
 - NAR
 - DCI
- Configuration registers—32-bit double word aligned configuration registers (byte and halfword access not supported)
- SPU actions—SPU block triggers various actions based on the derived events, these actions are defined in [Section 69.2, SPU actions](#)

69.2 SPU actions

Sequence actions describe what is triggered. A sequence state may only have a GoTo action to determine the next sequence state. The complete set of actions are as follows:

- Start/stop trace for a source
- Start/stop/increment counter/timer counter/timer[0–15]
- Reset counter/timer[0–15]
- Capture counter/timer Values[0–15]—place the specified counter/timer value into the trace stream for counters[0–15]
- Halt1—single signal (EVTI0) that can be used to halt a pre-configured DCI group1
- Halt2—single signal (EVTI1) that can be used to halt a pre-configured DCI group2
- Generate a watchpoint trigger for a respective ID field the NXMC (global timestamp must be enabled from the NAR)
- Capture the global timebase and place it into the trace stream with a watchpoint ID field of respective watchpoints at the NXMC (send TIMESTAMPx where x is related to an watchpoint ID field of NXMC only)
- Single interrupt for INTC to interrupt a CPU—de-assert (cleared) via JTAGM]
- Generate a pulse on the EVTO pin—output pulse over EVTO1 and EVTO2
- Start or Stop Performance Counters (PMC) for a CPU source—Start/Stop CPUn
- Optional trace group 1 (Start/Stop Trace a pre-configured set of trace signals)—four (DTM,PTM,WTM,OTM) independent start/stop trace groups can be formed. One or more clients (CPUn and Nexus) can be part of a particular group.

Note: GoTo is not part of the Action list described above but it is implemented in the state logic to move from one state to another. If there is nothing defined for GoTo, it signifies the end of the sequence.

In the condition where a state (State n) updates a counter and the next state in the sequence (State n+1) evaluates the value of the same counter (for example, to compare if the counter is less-than a certain value), the update to the counter (from State n) is not applied before the counter evaluation in the next state (State n+1). This results in the old value of the counter being evaluated (in State n+1), rather than the desired updated value. This is due to hardware limitations of the SPU as the update to the counter does not complete until the clock cycle after the evaluation of the next state. This limitation is only applicable in Div1 mode and not in Div4 mode.

To workaround this hardware limitation, a delay should be inserted between the update to a counter and the evaluation of this counter. A dummy state can be used between the counter update (State n) and the counter evaluation (State n+1). A dummy state is a state with any input, that on both then/else conditions, the action proceeds to the next state (State n+1) where the counter evaluation occurs. The insertion of this dummy state provides sufficient time for the update of the counter to be applied, in time for the counter evaluation. The same outcome is also possible in a complex sequence with multiple states by scheduling states so that counter update and counter evaluation are not performed on consecutive states in a sequence.

69.3 Operation mode

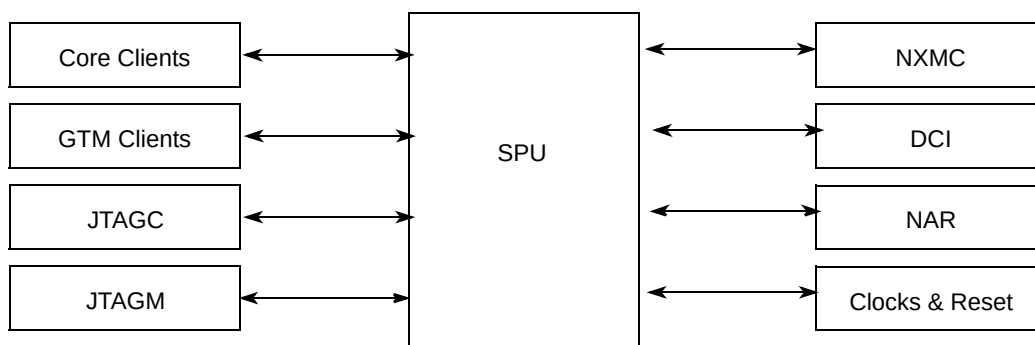
69.3.1 SPU enable/disable

By default, the SPU is in disabled mode. In this state, all events/inputs are disabled such that no counting or other actions occur. All the SPU control registers are accessible in this state. All the configurations must be done before enabling the SPU by programming the SPU Enable/Disable register.

69.4 SPU module interface

The SPU is connected to many clients such as the cores, GTM, system bus, NAR and Nexus clients. The SPU also interfaces to the DCI and the JTAG modules as shown in [Figure 1629](#).

Figure 1629. SPU module interface



69.5 Register definition

The SPU contains 32-bit aligned registers (byte and half-word access are not supported). All configuration registers in the SPU are JTAG registers such that these registers are accessible via JTAG interface only. Write access to reserved locations have no impact while read access to reserved locations always return 0. These registers have Nexus reset and JTAG reset control.

Note: The SPU does not check for correctness of the programmed values in the registers and programmers must ensure that correct values are being written.

[Table 1535](#) lists the memory mapped registers which control the operation of the SPU. There are two instructions for the SPU registers:

- Instruction to enable the SLU (first set of registers with offsets 1–0x49) registers is 4'h3
- Instruction to enable the Counter Control Unit (second set of registers with offsets 1–72) is 4'h2

Table 1535. SPU register summary

Offset (hex)	Register	Access	Reset Value	Location
1	Level1 Mux Selection 0 (L1SEL0)	R/W	32'h0	on page 2987
2	Level1 Mux Selection 1 (L1SEL1)	R/W	32'h0	on page 2987
3	Level1 Mux Selection 2 (L1SEL2)	R/W	32'h0	on page 2988
4	Level1 Mux Selection 3 (L1SEL3)	R/W	32'h0	on page 2988
5	Level1 Mux Selection 4 (L1SEL4)	R/W	32'h0	on page 2988
6	Level1 Mux Selection 5 (L1SEL5)	R/W	32'h0	on page 2989
7	Level1 Mux Selection 6 (L1SEL6)	R/W	32'h0	on page 2989
8	Level1 Mux Selection 7 (L1SEL7)	R/W	32'h0	on page 2989
9	Reserved for future use	—	—	—
A	Reserved for future use	—	—	—
B	Input Trigger Level Detection 2 (ITLD2)	R/W	32'h0	page 2990
C	Input Trigger Level Detection 3 (ITLD3)	R/W	32'h0	page 2990
D	CPU0 Processor Except Vector Prefix (C0PEVP)	R/W	32'h0	on page 2991
E	CPU1 Processor Except Vector Prefix (C1PEVP)	R/W	32'h0	on page 2991
F	CPU2 Processor Except Vector Prefix (C2PEVP)	R/W	32'h0	on page 2992
10	Reserved for future use	—	—	—
11	Reserved for future use	—	—	—
12	CPU0 Interrupt Priority Selection (C0PIS)	R/W	32'h0	on page 2993
13	CPU1 Interrupt Priority Selection (C1PIS)	R/W	32'h0	on page 2993
14	CPU2 Interrupt Priority Selection (C2PIS)	R/W	32'h0	on page 2994
15	Reserved for future use	—	—	—
16	Reserved for future use	—	—	—
17	Level2 Mux State0 Selection 0 (L20SEL0)	R/W	32'h0	on page 2995
18	Level2 Mux State0 Selection 1 (L20SEL1)	R/W	32'h0	on page 2997
19	Level2 Mux State0 Selection 2 (L20SEL2)	R/W	32'h0	on page 2998
1A	Level2 Mux State0 Selection 3 (L20SEL3)	R/W	32'h0	on page 2999
1B	Level2 Mux State1 Selection 0 (L21SEL0)	R/W	32'h0	on page 2995
1C	Level2 Mux State1 Selection 1 (L21SEL1)	R/W	32'h0	on page 2997
1D	Level2 Mux State1 Selection 2 (L21SEL2)	R/W	32'h0	on page 2998
1E	Level2 Mux State1 Selection 3 (L21SEL3)	R/W	32'h0	on page 2999
1F	Level2 Mux State2 Selection 0 (L22SEL0)	R/W	32'h0	on page 2995
20	Level2 Mux State2 Selection 1 (L22SEL1)	R/W	32'h0	on page 2997
21	Level2 Mux State2 Selection 2 (L22SEL2)	R/W	32'h0	on page 2998
22	Level2 Mux State2 Selection 3 (L22SEL3)	R/W	32'h0	on page 2999

Table 1535. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value	Location
23	Level2 Mux State3 Selection 0 (L23SEL0)	R/W	32'h0	on page 2995
24	Level2 Mux State3 Selection 1 (L23SEL1)	R/W	32'h0	on page 2997
25	Level2 Mux State3 Selection 2 (L23SEL2)	R/W	32'h0	on page 2998
26	Level2 Mux State3 Selection 3 (L23SEL3)	R/W	32'h0	on page 2999
27	Level2 Mux State4 Selection 0 (L24SEL0)	R/W	32'h0	on page 2995
28	Level2 Mux State4 Selection 1 (L24SEL1)	R/W	32'h0	on page 2997
29	Level2 Mux State4 Selection 2 (L24SEL2)	R/W	32'h0	on page 2998
2A	Level2 Mux State4 Selection 3 (L24SEL3)	R/W	32'h0	on page 2999
2B	Level2 Mux State5 Selection 0 (L25SEL0)	R/W	32'h0	on page 2995
2C	Level2 Mux State5 Selection 1 (L25SEL1)	R/W	32'h0	on page 2997
2D	Level2 Mux State5 Selection 2 (L25SEL2)	R/W	32'h0	on page 2998
2E	Level2 Mux State6 Selection 3 (L25SEL3)	R/W	32'h0	on page 2999
2F	Level2 Mux State6 Selection 0 (L26SEL0)	R/W	32'h0	on page 2995
30	Level2 Mux State6 Selection 1 (L26SEL1)	R/W	32'h0	on page 2997
31	Level2 Mux State6 Selection 2 (L26SEL2)	R/W	32'h0	on page 2998
32	Level2 Mux State6 Selection 3 (L26SEL3)	R/W	32'h0	on page 2999
33	Level2 Mux State7 Selection 0 (L27SEL0)	R/W	32'h0	on page 2995
34	Level2 Mux State7 Selection 1 (L27SEL1)	R/W	32'h0	on page 2997
35	Level2 Mux State7 Selection 2 (L27SEL2)	R/W	32'h0	on page 2998
36	Level2 Mux State7 Selection 3 (L27SEL3)	R/W	32'h0	on page 2999
37	Input/Output Inversion Control State0 (IOIC0)	R/W	32'h0	on page 3001
38	Input/Output Inversion Control State1 (IOIC1)	R/W	32'h0	on page 3001
39	Input/Output Inversion Control State2 (IOIC2)	R/W	32'h0	on page 3001
3A	Input/Output Inversion Control State3 (IOIC3)	R/W	32'h0	on page 3001
3B	Input/Output Inversion Control State4 (IOIC4)	R/W	32'h0	on page 3001
3C	Input/Output Inversion Control State5 (IOIC5)	R/W	32'h0	on page 3001
3D	Input/Output Inversion Control State6 (IOIC6)	R/W	32'h0	on page 3001
3E	Input/Output Inversion Control State7 (IOIC7)	R/W	32'h0	on page 3001
3F	Sequence Control (SCTRL)	R/W	32'h0	on page 3002
40	State0 Goto Control (S0GC)	R/W	32'h0	on page 3005
41	State1 Goto Control (S1GC)	R/W	32'h0	on page 3005
42	State2 Goto Control (S2GC)	R/W	32'h0	on page 3005
43	State3 Goto Control (S3GC)	R/W	32'h0	on page 3005
44	State4 Goto Control (S4GC)	R/W	32'h0	on page 3005

Table 1535. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value	Location
45	State5 Goto Control (S5GC)	R/W	32'h0	on page 3005
46	State6 Goto Control (S6GC)	R/W	32'h0	on page 3005
47	State7 Goto Control (S7GC)	R/W	32'h0	on page 3005
48	SLU Status (SS)	R/W	32'h0	on page 3006
49	SPU Enable (SE)	R/W	32'h0	on page 3009
1	State0 True Action 0 (S0TA0)	R/W	32'h0	on page 3012
2	State0 True Action 1 (S0TA1)	R/W	32'h0	on page 3013
3	State1 True Action 0 (S1TA0)	R/W	32'h0	on page 3012
4	State1 True Action 1 (S1TA1)	R/W	32'h0	on page 3013
5	State2 True Action 0 (S2TA0)	R/W	32'h0	on page 3012
6	State2 True Action 1 (S2TA1)	R/W	32'h0	on page 3013
7	State3 True Action 0 (S3TA0)	R/W	32'h0	on page 3012
8	State3 True Action 1 (S3TA1)	R/W	32'h0	on page 3013
9	State4 True Action 0 (S4TA0)	R/W	32'h0	on page 3012
A	State4 True Action 1 (S4TA1)	R/W	32'h0	on page 3013
B	State5 True Action 0 (S5TA0)	R/W	32'h0	on page 3012
C	State5 True Action 1 (S5TA1)	R/W	32'h0	on page 3013
D	State6 True Action 0 (S6TA0)	R/W	32'h0	on page 3012
E	State6 True Action 1 (S6TA1)	R/W	32'h0	on page 3013
F	State7 True Action 0 (S7TA0)	R/W	32'h0	on page 3012
10	State7 True Action 1 (S7TA1)	R/W	32'h0	on page 3013
11	State0 False Action 0 (S0FA0)	R/W	32'h0	on page 3014
12	State0 False Action 1 (S0FA1)	R/W	32'h0	on page 3015
13	State1 False Action 0 (S1FA0)	R/W	32'h0	on page 3014
14	State1 False Action 1 (S1FA1)	R/W	32'h0	on page 3015
15	State2 False Action 0 (S2FA0)	R/W	32'h0	on page 3014
16	State2 False Action 1 (S2FA1)	R/W	32'h0	on page 3015
17	State3 False Action 0 (S3FA0)	R/W	32'h0	on page 3014
18	State3 False Action 1 (S3FA1)	R/W	32'h0	on page 3015
19	State4 False Action 0 (S4FA0)	R/W	32'h0	on page 3014
1A	State4 False Action 1 (S4FA1)	R/W	32'h0	on page 3015
1B	State5 False Action 0 (S5FA0)	R/W	32'h0	on page 3014
1C	State5 False Action 1 (S5FA1)	R/W	32'h0	on page 3015
1D	State6 False Action 0 (S6FA0)	R/W	32'h0	on page 3014

Table 1535. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value	Location
1E	State6 False Action 1 (S6FA1)	R/W	32'h0	on page 3015
1F	State7 False Action 0 (S7FA0)	R/W	32'h0	on page 3014
20	State7 False Action 1 (S7FA1)	R/W	32'h0	on page 3015
21	DTM Trace Group Configuration (DTMTGC)	R/W	32'h0	on page 3016
22	PTM Trace Group Configuration (PTMTGC)	R/W	32'h0	on page 3016
23	OTM Trace Group Configuration (OTMTGC)	R/W	32'h0	on page 3016
24	WTM Trace Group Configuration (WTMTGC)	R/W	32'h0	on page 3016
25	Interrupt Status (INTS)	R/W	32'h0	on page 3017
26	Counter Control 0 (CTRL0)	R/W	32'h0	on page 3018
27	Counter Control 1 (CTRL1)	R/W	32'h0	on page 3018
28	Counter Control 2 (CTRL2)	R/W	32'h0	on page 3018
29	Counter Control 3 (CTRL3)	R/W	32'h0	on page 3018
2A	Counter Control 4 (CTRL4)	R/W	32'h0	on page 3018
2B	Counter Control 5 (CTRL5)	R/W	32'h0	on page 3018
2C	Counter Control 6 (CTRL6)	R/W	32'h0	on page 3018
2D	Counter Control 7 (CTRL7)	R/W	32'h0	on page 3018
2E	Counter Control 8 (CTRL8)	R/W	32'h0	on page 3018
2F	Counter Control 9 (CTRL9)	R/W	32'h0	on page 3018
30	Counter Control 10 (CTRL10)	R/W	32'h0	on page 3018
31	Counter Control 11 (CTRL11)	R/W	32'h0	on page 3018
32	Counter Control 12 (CTRL12)	R/W	32'h0	on page 3018
33	Counter Control 13 (CTRL13)	R/W	32'h0	on page 3018
34	Counter Control 14 (CTRL14)	R/W	32'h0	on page 3018
35	Counter Control 15 (CTRL15)	R/W	32'h0	on page 3018
36	Counter Compare 0 (CCMP0)	R/W	32'h0	on page 3019
37	Counter Compare 1 (CCMP1)	R/W	32'h0	on page 3019
38	Counter Compare 2 (CCMP2)	R/W	32'h0	on page 3019
39	Counter Compare 3 (CCMP3)	R/W	32'h0	on page 3019
3A	Counter Compare 4 (CCMP4)	R/W	32'h0	on page 3019
3B	Counter Compare 5 (CCMP5)	R/W	32'h0	on page 3019
3C	Counter Compare 6 (CCMP6)	R/W	32'h0	on page 3019
3D	Counter Compare 7 (CCMP7)	R/W	32'h0	on page 3019
3E	Counter Compare 8 (CCMP8)	R/W	32'h0	on page 3019
3F	Counter Compare 9 (CCMP9)	R/W	32'h0	on page 3019

Table 1535. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value	Location
40	Counter Compare 10 (CCMP10)	R/W	32'h0	on page 3019
41	Counter Compare 11 (CCMP11)	R/W	32'h0	on page 3019
42	Counter Compare 12 (CCMP12)	R/W	32'h0	on page 3019
43	Counter Compare 13 (CCMP13)	R/W	32'h0	on page 3019
44	Counter Compare 14 (CCMP14)	R/W	32'h0	on page 3019
45	Counter Compare 15 (CCMP15)	R/W	32'h0	on page 3019
46	Counter Compare Status (CCAPS)	R/W	32'h0	on page 3020
47	Counter Overflow Status (COS)	R/W	32'h0	on page 3020
48	Counter Capture Status (CCOMS)	R/W	32'h0	on page 3021

69.5.1 Register descriptions

69.5.1.1 Level1 input Mux configurations registers

A set of input Mux control registers is used to narrow down the large number of SPU watchpoints to a manageable set. There are sixty-four 8×1 (eight inputs and one output) multiplexers. Three bits per Mux are used to select one input out of 8 inputs. 8 Muxes are configured with each 32-bit register; in other words, 8 inputs out of 64 for each register. This provides the flexibility to the programmer to route any input trigger to any state input. The detail of this selection is shown in [Figure 1667](#).

Note: Users cannot program MUX 63 in L1SEL7 (the 64th mux). All eight inputs of the 63rd mux are tied to high and the output is always high (1'b1).

69.5.1.1.1 Level1 Mux Selection 0 (L1SEL0)

[Figure 1630](#) shows the format of the L1SEL0 register. The values for the fields of the L1SEL0 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x01

Access: User read/write

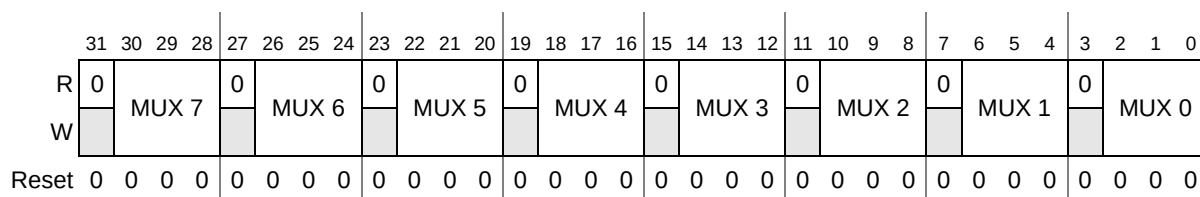


Figure 1630. L1SEL0 register format

69.5.1.1.2 Level1 Mux Selection 1 (L1SEL1)

[Figure 1631](#) shows the format of the L1SEL1 register. The values for the fields of the L1SEL1 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset 0x02																Access: User read/write																								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
R	0	MUX 15				0	MUX 14				0	MUX 13				0	MUX 12				0	MUX 11				0	MUX 10				0	MUX 9				0	MUX 8			
W																																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

Figure 1631. L1SEL1 register format

69.5.1.1.3 Level1 Mux Selection 2 (L1SEL2)

[Figure 1632](#) shows the format of the L1SEL2 register. The values for the fields of the L1SEL2 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset 0x03																Access: User read/write																								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
R	0	MUX 23				0	MUX 22				0	MUX 21				0	MUX 20				0	MUX 19				0	MUX 18				0	MUX 17				0	MUX 16			
W																																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						

Figure 1632. L1SEL2 register format

69.5.1.1.4 Level1 Mux Selection 3 (L1SEL3)

[Figure 1633](#) shows the format of the L1SEL3 register. The values for the fields of the L1SEL3 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset 0x04																Access: User read/write																								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
R	0	MUX 31				0	MUX 30				0	MUX 29				0	MUX 28				0	MUX 27				0	MUX 26				0	MUX 25				0	MUX 24			
W																																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

Figure 1633. L1SEL3 register format

69.5.1.1.5 Level1 Mux Selection 4 (L1SEL4)

[Figure 1634](#) shows the format of the L1SEL4 register. The values for the fields of the L1SEL4 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset 0x05

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0				0				0				0				0				0				0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1634. L1SEL4 register format

69.5.1.1.6 Level1 Mux Selection 5 (L1SEL5)

[Figure 1635](#) shows the format of the L1SEL5 register. The values for the fields of the L1SEL5 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset 0x06

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0				0				0				0				0				0				0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1635. L1SEL5 register format

69.5.1.1.7 Level1 Mux Selection 6 (L1SEL6)

[Figure 1636](#) shows the format of the L1SEL6 register. The values for the fields of the L1SEL6 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset 0x07

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0				0				0				0				0				0				0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1636. L1SEL6 register format

69.5.1.1.8 Level1 Mux Selection 7 (L1SEL7)

[Figure 1637](#) shows the format of the L1SEL7 register. The values for the fields of the L1SEL7 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured. All eight inputs of MUX 63 are tied to logic high (1'b1) and output is always high.

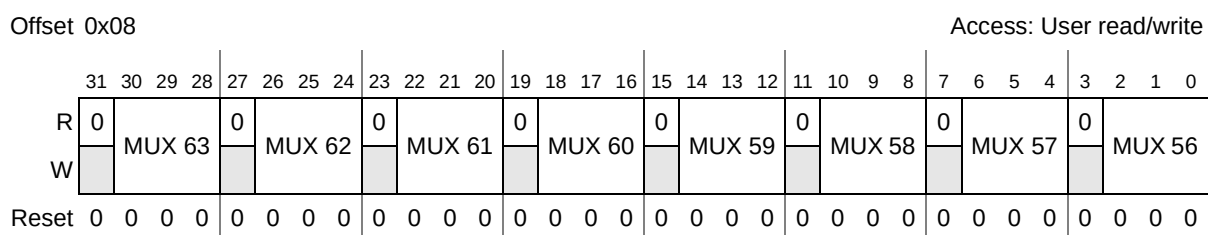
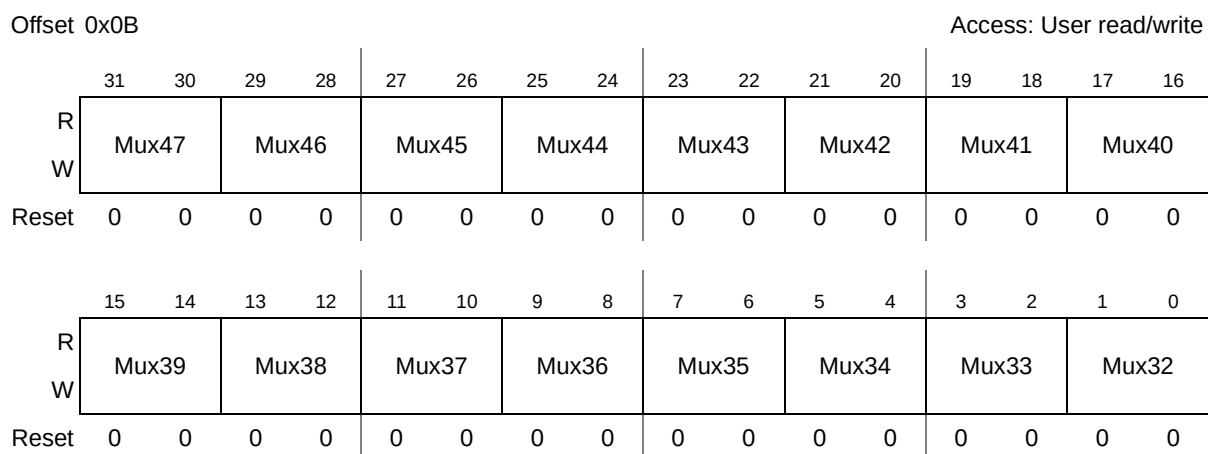


Figure 1637. L1SEL7 register format

69.5.1.1.9 Input triggers level detection control 2 (ITLD2)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and therefore these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. [Figure 1638](#) shows the format of the ITLD2 register. All of the MUX fields in the ITLD2 register have the same description as shown in [Table 1536](#).

Figure 1638. ITLD2 register format



The ITLD registers Mux field is described in [Table 1536](#).

Table 1536. ITLD register field descriptions

Field	Description
Mux*	Mux Output Signal Detection. 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

69.5.1.1.10 Input triggers level detection control 3 (ITLD3)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and so these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. [Figure 1639](#) shows the format of the ITLD3 register. All of the MUX fields in the ITLD3 register have the

same description as shown in [Table 1536](#).

Figure 1639. ITLD3 register format

Offset 0x0C												Access: User read/write								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
R	Mux63				Mux62				Mux61				Mux 60				Mux59			
W	Mux58				Mux57				Mux56				Mux55				Mux54			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
R	Mux55				Mux54				Mux53				Mux52				Mux51			
W	Mux50				Mux49				Mux48				Mux47				Mux46			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

69.5.1.1.11 CPU0 processor exception vector prefix (C0PEVP)

The SPU captures the exception vector from CPU0 when the exception enable signal is

Offset 0x0D																Access: User read/write																
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU0															
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1640. C0PEVP register format

The C0PEVP register is described in [Table 1537](#).

Table 1537. C0PEVP register field descriptions

Field	Description
15:0 CPU0	CPU0 processor exception vector. Program this field with one of the vector values from Table 1564 to select the corresponding processor exception. <i>Note: This field must use the byte swapped values in the CnPEVP register CPUUn field value column in Table 1564.</i>
31:16	Reserved

69.5.1.1.12 CPU1 processor exception vector prefix (C1PEVP)

The SPU captures the exception vector from CPU1 when the exception enable signal is asserted. The exception vector decoding is defined in [Table 1564](#). The SPU compares this captured vector with the value in the C1PEVP register. [Figure 1641](#) shows the format of the C1PEVP register.

Offset 0x0E

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU1															
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1641. C1PEVP register format

The C1PEVP register is described in [Table 1538](#).

Table 1538. C1PEVP register field descriptions

Field	Description
15:0 CPU1	CPU1 processor exception vector. Program this field with one of the vector values from Table 1564 to select the corresponding processor exception. <i>Note: This field must use the byte swapped values in the CnPEVP register CPU_n field value column in Table 1564.</i>
31:16	Reserved

69.5.1.1.13 CPU2 processor exception vector prefix (C2PEVP)

The SPU captures the exception vector from CPU2 when the exception enable signal is asserted. The exception vector decoding is defined in [Table 1564](#). The SPU compares this captured vector with the value in the C2PEVP register. [Figure 1642](#) shows the format of the C2PEVP register.

Offset 0x0F

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU2															
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1642. C2PEVP register format

The C2PEVP register is described in [Table 1539](#).

Table 1539. C2PEVP register field descriptions

Field	Description
15:0 CPU2	CPU2 processor exception vector. Program this field with one of the vector values from Table 1564 to select the corresponding processor exception. <i>Note: This field must use the byte swapped values in the CnPEVP register CPU_n field value column in Table 1564.</i>
31:16	Reserved

69.5.1.1.14 CPU0 interrupt priority selection (C0PIS)

[Figure 1643](#) shows the format of the C0PIS register. The SPU compares the input CPR vector from CPU0 with the value programmed to the C0PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

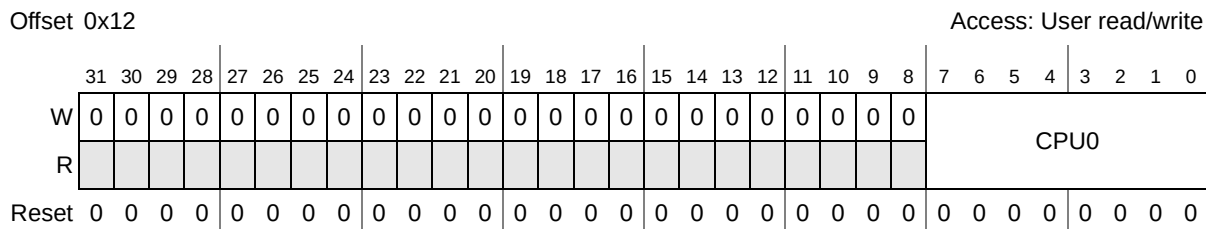


Figure 1643. C0PIS register format

The C0PIS register is described in [Table 1540](#).

Table 1540. C0PIS register field descriptions

Field	Description
7:0 CPU0	CPU0 Priority Interrupt Selection.
31:8	Reserved

69.5.1.1.15 CPU1 interrupt priority selection (C1PIS)

[Figure 1644](#) shows the format of the C1PIS register. The SPU compares the input CPR vector from CPU1 with the value programmed to the C1PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

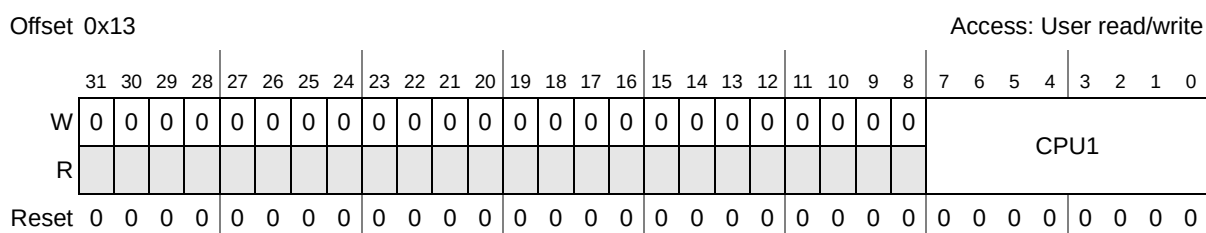


Figure 1644. C1PIS register format

The C1PIS register is described in [Table 1541](#).

Table 1541. C1PIS register field descriptions

Field	Description
7:0 CPU1	CPU1 Priority Interrupt Selection.
31:8	Reserved

69.5.1.1.16 CPU2 interrupt priority selection (C2PIS)

[Figure 1645](#) shows the format of the C2PIS register. The SPU compares the input CPR vector from CPU2 with the value programmed to the C2PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

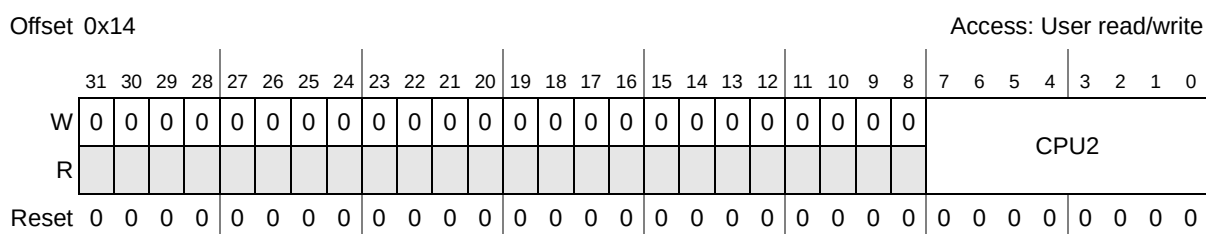


Figure 1645. C2PIS register format

The C2PIS register is described in [Table 1542](#).

Table 1542. C2PIS register field descriptions

Field	Description
7:0 CPU2	CPU2 Priority Interrupt Selection.
31:8	Reserved

69.5.1.2 Level2 input Mux configurations registers

At level1, the set of input Mux control registers (described in [Section 69.5.1.1, Level1 input Mux configurations registers](#)) narrows down the large number of SPU watchpoints to a manageable set using sixty-four 8×1 multiplexers. At Level2, 16 inputs are selected for each SLU from the 64 level1 outputs using 16 64×1 multiplexers. Six bits per Mux select one input out of 64 (in fact 63) inputs. Each 32-bit register configures the AND gates of one state; therefore, eight registers are used to configure inputs for the 8 SLUs. All the 63 inputs are static for all the eight states. Any 16 inputs can be selected from the 63 input triggers for a state.

The detail selection is shown in the [Figure 1667](#) for State0. Similar logic is provided for States 1–7.

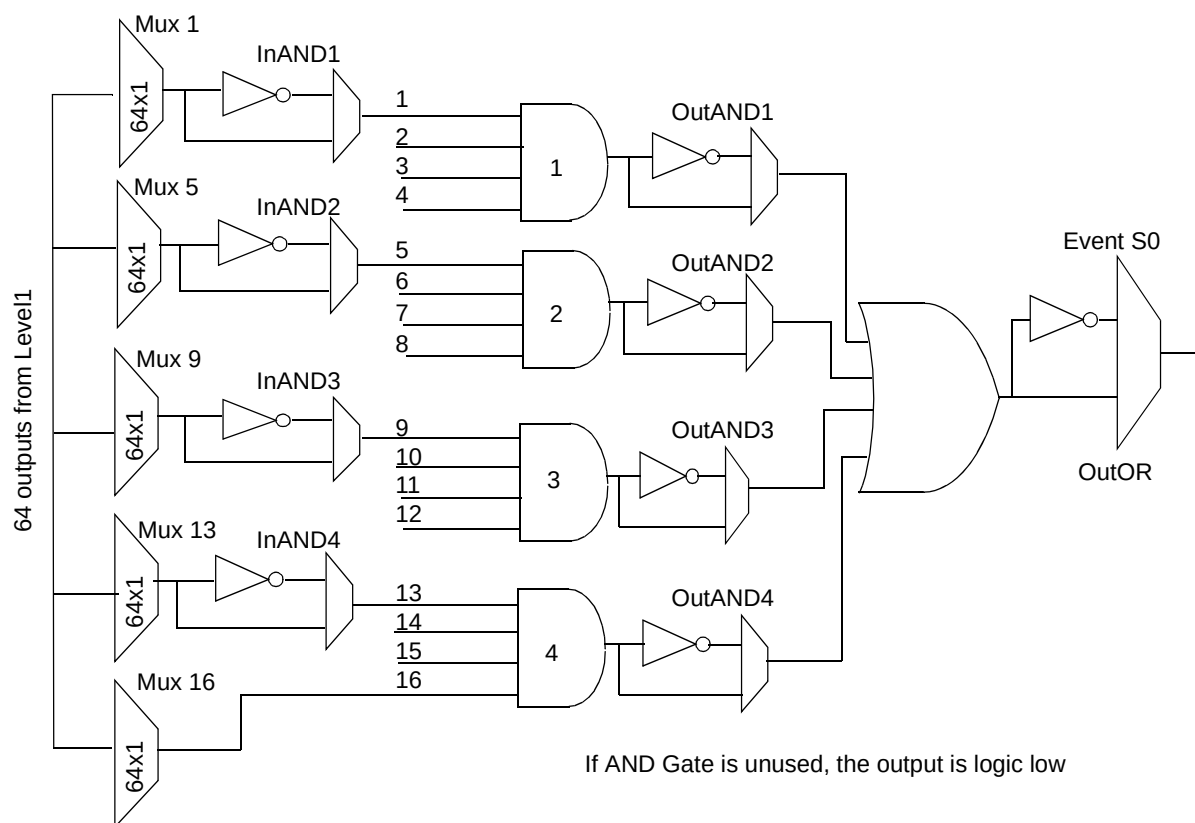


Figure 1646. Level2 Mux AND gates input pin routing for State0

Note: If any SLU AND gate is unused, then its output is logic low (1'b0). Thus, if all four inputs of a particular AND gate are at the default value 1'b1 (none of the inputs have been routed from the Level 2 mux to the inputs of AND gates), then the output of that AND gate is logic low. This provides protection to avoid the masking of other inputs to the OR gate.

69.5.1.2.1 Level2 Mux state n selection 0 (L2nSEL0)

[Figure 1647](#) shows the format of the L2nSEL0 register, where n, the state number, is equal to 0–7.

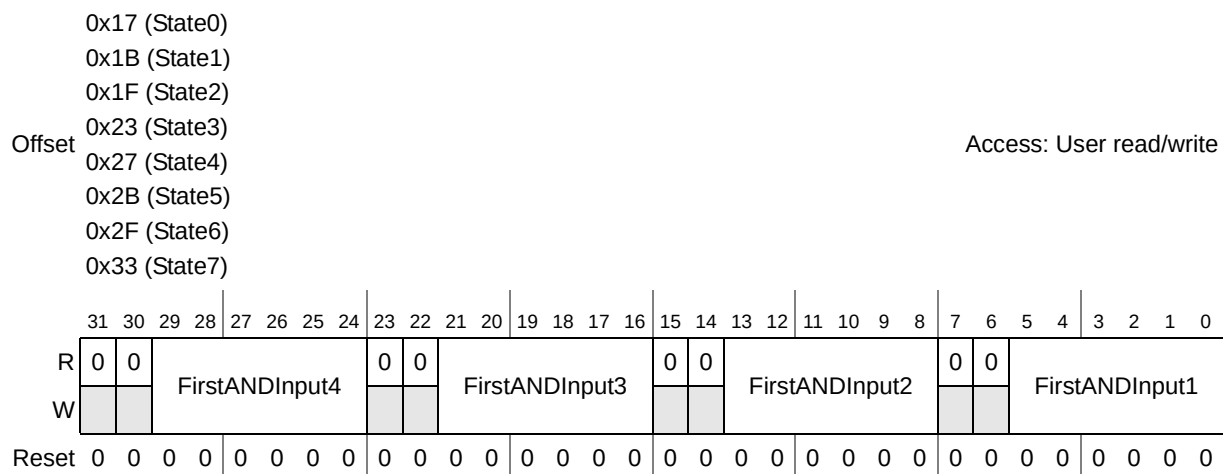


Figure 1647. L2nSEL0 register format

The L2nSEL0 register fields are described in [Table 1543](#).

Table 1543. L2nSEL0 register field descriptions

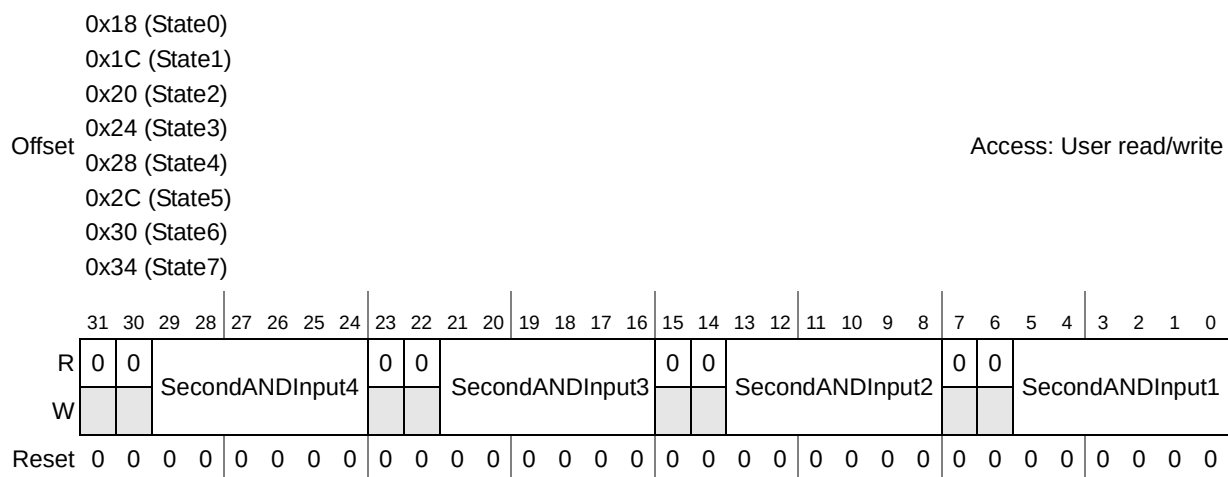
Field	Description
5:0 FirstANDInput1	First AND Gate Input1 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
7:6	Reserved. Read returns 0.
13:8 FirstANDInput2	First AND Gate Input 2 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
15:14	Reserved. Read returns 0.
21:16 ANDInput3	First AND Gate Input 3 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
23:22	Reserved. Read returns 0.

Table 1543. L2nSEL0 register field descriptions (continued)

Field	Description
29:24 FirstANDInput4	First AND Gate Input 4 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
31:30	Reserved. Read returns 0.

69.5.1.2.2 Level2 Mux state n selection 1 (L2nSEL1)

[Figure 1648](#) shows the format of the L2nSEL1 register, where n, the state number, is equal to 0–7.

**Figure 1648. L2nSEL1 register format**

The L2nSEL1 register fields are described in [Table 1544](#).

Table 1544. L2nSEL1 register field descriptions

Field	Description
5:0 SecondANDInput1	Second AND Gate Input1 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
7:6	Reserved. Read returns 0.

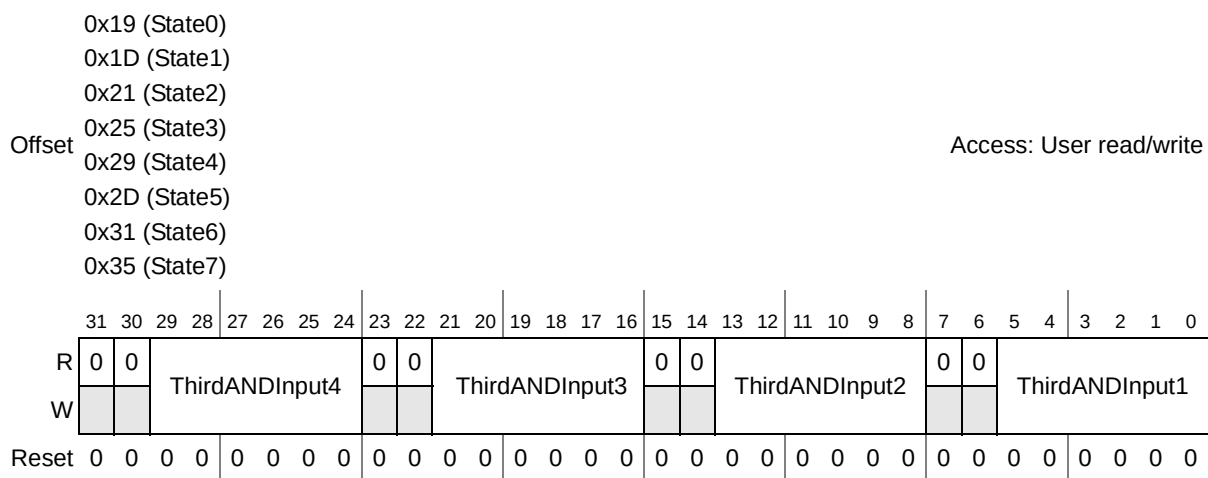
Table 1544. L2nSEL1 register field descriptions (continued)

Field	Description
13:8 SecondANDInput2	Second AND Gate Input 2 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
15:14	Reserved. Read returns 0.
21:16 SecondANDInput3	Second AND Gate Input 3 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
23:22	Reserved. Read returns 0.
29:24 SecondANDInput4	Second AND Gate Input 4 Selection. 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
31:30	Reserved. Read returns 0.

69.5.1.2.3 Level2 Mux state n selection 2 (L2nSEL2)

Figure 1649 shows the format of the L2nSEL2 register, where n, the state number, is equal to 0–7.

Figure 1649. L2nSEL2 register format



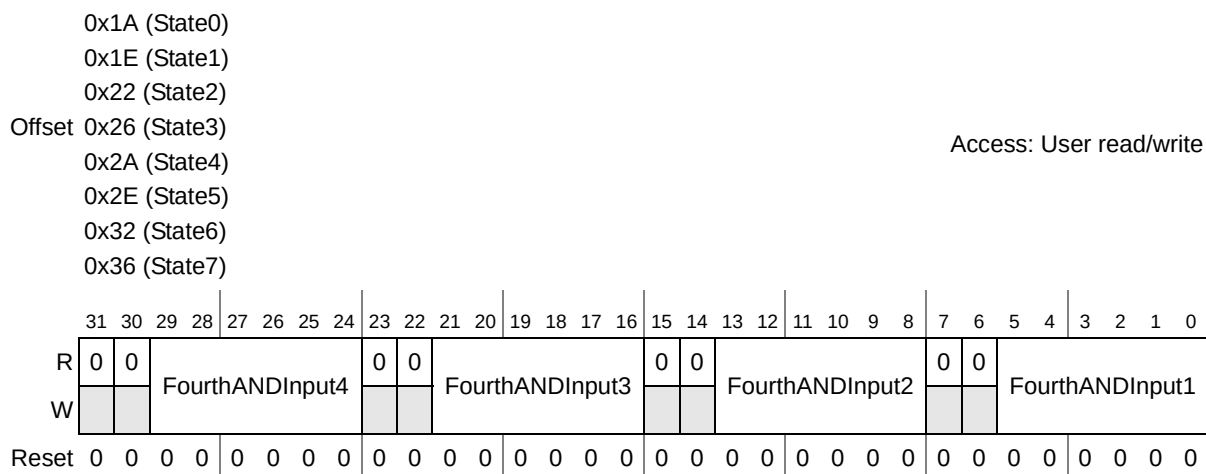
The L2nSEL2 register fields are described in [Table 1545](#).

Table 1545. L2nSEL2 register field descriptions

Field	Description
5:0 ThirdANDInput1	Third AND Gate Input1 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
7:6	Reserved. Read returns 0.
13:8 ThirdANDInput2	Third AND Gate Input 2 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
15:14	Reserved. Read returns 0.
21:16 ThirdANDInput3	Third AND Gate Input 3 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
23:22	Reserved. Read returns 0.
29:24 ThirdANDInput4	Third AND Gate Input 4 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
31:30	Reserved. Read returns 0.

69.5.1.2.4 Level2 Mux state n selection 3 (L2nSEL3)

[Figure 1650](#) shows the format of the L2nSEL3 register, where n, the state number, is equal to 0–7.

**Figure 1650. L2nSEL3 register format**

The L2nSEL3 register fields are described in [Table 1546](#).

Table 1546. L2nSEL3 register field descriptions

Field	Description
5:0 FourthANDInput1	Fourth AND Gate Input1 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
7:6	Reserved. Read returns 0.
13:8 FourthANDInput2	Fourth AND Gate Input 2 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
15:14	Reserved. Read returns 0.
21:16 FourthANDInput3	Fourth AND Gate Input 3 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
23:22	Reserved. Read returns 0.

Table 1546. L2nSEL3 register field descriptions (continued)

Field	Description
29:24 FourthANDInput4	Fourth AND Gate Input 4 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output ... 111110Level1 MUX 61 output 111111Level1 MUX 62 output
31:30	Reserved. Read returns 0.

69.5.1.3 Sequence unit registers

69.5.1.3.1 Input/output Inversion control state n (IOICn)

An input control register is provided for each event generated by the SLU. This register is programmed to specify how the input signals to each SLU are combined to generate the resulting event for that particular sequence. In total, there are 16 inputs for each state and only one (the first) of the inputs can be used with or without inversion. The input can be left unused if not required for that state by selecting 000000 (No input selected option) in the L2nSEL registers. The output of each AND gate has the optional inversion control as well. The output can be routed directly or with inversion from each AND gate to the input of OR gate. [Figure 1651](#) shows the format of the IOICn register, where n, the state number, is equal to 0–7.

Figure 1651. IOICn register format

0x37 (State0)																	
0x38 (State1)																	
0x39 (State2)																	
Offset 0x3A (State3)																	
0x3B (State4)																	
0x3C (State5)																	
0x3D (State6)																	
0x3E (State7)																	
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0	0	0	0	0	0	0	0	OutOR	OutAND4	OutAND3	OutAND2	OutAND1	InAND4	InAND3	InAND2	InAND1
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The IOICn register fields are described in [Table 1547](#).

Table 1547. IOICn register field descriptions

Field	Description
0 InAND1	Input 1 AND Gate 1 Control. 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
1 InAND2	Input 1 AND Gate 2 Control. 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
2 InAND3	Input 1 AND Gate 3 Control. 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
3 InAND4	Input 1 AND Gate 4 Control. 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
4 OutAND1	Output AND Gate 1 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
5 OutAND2	Output AND Gate 2 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
6 OutAND3	Output AND Gate 3 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
7 OutAND4	Output of AND Gate 4 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
8 OutOR	Output OR GATE Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
9:31	Reserved. Read returns 0.

69.5.1.3.2 Sequence control (SCTRL)

This configuration register is provided to enable the sequence and to select the states that are part of a particular sequence. There can be at most four active sequences. Each sequence can have one state or a maximum of eight states. However, each state can only be used in one unique sequence. In other words, if one state has been selected for a sequence, then this state cannot be used for any other defined sequence. Each sequence has a start state and end state. [Figure 1652](#) shows the format of the SCTRL register.

Figure 1652. SCTRL register format

Offset 0x3F																Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	ST2			ST1			ST0	0	ST2			ST1			ST0			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	0	ST2			ST1			ST0	0	ST2			ST1			ST0			
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

The SCTRL register fields are described in [Table 1548](#).

Table 1548. SCTRL register field descriptions

Field	Description
0 ST0	Sequence1 Enable/Disable ⁽¹⁾ . Once enabled, this bit is disabled at the end of the Sequence 1. 0 Sequence 1 is disabled 1 Sequence 1 is enabled
3:1 ST1	Sequence 1 Start State. 000 State 0 (Always)
6:4 ST2	Sequence 1 End State. 000 State 0 is the Stop state for sequence 1 001 State 1 is the Stop state for sequence 1 010 State 2 is the Stop state for sequence 1 011 State 3 is the Stop state for sequence 1 100 State 4 is the Stop state for sequence 1 101 State 5 is the Stop state for sequence 1 110 State 6 is the Stop state for sequence 1 111 State 7 is the Stop state for sequence 1
7	Reserved. Read returns 0.
8 ST0	Sequence2 Enable/Disable ⁽¹⁾ . Once enabled, this bit is disabled at the end of the Sequence 2. 0 Sequence 2 is disabled 1 Sequence 2 is enabled
11:9 ST1	Sequence 2 Start State. ⁽²⁾ 000 State 0 is the Start state for sequence 2 001 State 1 is the Start state for sequence 2 010 State 2 is the Start state for sequence 2 011 State 3 is the Start state for sequence 2 100 State 4 is the Start state for sequence 2 101 State 5 is the Start state for sequence 2 110 State 6 is the Start state for sequence 2 111 State 7 is the Start state for sequence 2

Table 1548. SCTRL register field descriptions (continued)

Field	Description
14:12 ST2	Sequence 2 End State. 000 State 0 is the Stop state for sequence 2 001 State 1 is the Stop state for sequence 2 010 State 2 is the Stop state for sequence 2 011 State 3 is the Stop state for sequence 2 100 State 4 is the Stop state for sequence 2 101 State 5 is the Stop state for sequence 2 110 State 6 is the Stop state for sequence 2 111 State 7 is the Stop state for sequence 2
15	Reserved. Read returns 0.
16 ST0	Sequence 3 Enable/Disable ⁽¹⁾ . Once enabled, this bit is disabled at the end of the Sequence 3. 0 Sequence 3 is disabled 1 Sequence 3 is enabled
19:17 ST1	Sequence 3 Start State. ⁽²⁾ 000 State 0 is the Start state for sequence 3 001 State 1 is the Start state for sequence 3 010 State 2 is the Start state for sequence 3 011 State 3 is the Start state for sequence 3 100 State 4 is the Start state for sequence 3 101 State 5 is the Start state for sequence 3 110 State 6 is the Start state for sequence 3 111 State 7 is the Start state for sequence 3
22:20 ST2	Sequence 3 End State. 000 State 0 is the Stop state for sequence 3 001 State 1 is the Stop state for sequence 3 010 State 2 is the Stop state for sequence 3 011 State 3 is the Stop state for sequence 3 100 State 4 is the Stop state for sequence 3 101 State 5 is the Stop state for sequence 3 110 State 6 is the Stop state for sequence 3 111 State 7 is the Stop state for sequence 3
23	Reserved. Read returns 0.
24 ST0	Sequence 4 Enable/Disable ⁽¹⁾ . Once enabled, this bit is disabled at the end of the Sequence 4. 0 Sequence 4 is disabled 1 Sequence 4 is enabled
27:25 ST1	Sequence 4 Start State. ⁽²⁾ 000 State 0 is the Start state for sequence 4 001 State 1 is the Start state for sequence 4 010 State 2 is the Start state for sequence 4 011 State 3 is the Start state for sequence 4 100 State 4 is the Start state for sequence 4 101 State 5 is the Start state for sequence 4 110 State 6 is the Start state for sequence 4 111 State 7 is the Start state for sequence 4

Table 1548. SCTRL register field descriptions (continued)

Field	Description
30:28 ST2	Sequence 4 End State. 000 State 0 is the Stop state for sequence 4 001 State 1 is the Stop state for sequence 4 010 State 2 is the Stop state for sequence 4 011 State 3 is the Stop state for sequence 4 100 State 4 is the Stop state for sequence 4 101 State 5 is the Stop state for sequence 4 110 State 6 is the Stop state for sequence 4 111 State 7 is the Stop state for sequence 4
31	Reserved. Read returns 0.

- Multiple active sequences such as Sequence 2, Sequence 3 and Sequence 4 must be programmed with Div4 option only.
- Ideally, the start state for a sequence should be the state immediately after the stop state for the previous sequence. For example, if sequence 1 starts at state 0 and ends at state3, sequence 2 should start at state4.

69.5.1.3.3 State n goto control (SnGC)

SLU states are linked to each other based on if-then-else conditions. There are two conditions: true and false. A SLU state can remain in the same state or jump to any other state. The state goto control register is programmed to specify the GoTo a particular state from current state.

[Figure 1653](#) shows the format of the SnGC register, where n, the state number, is equal to 0–7.

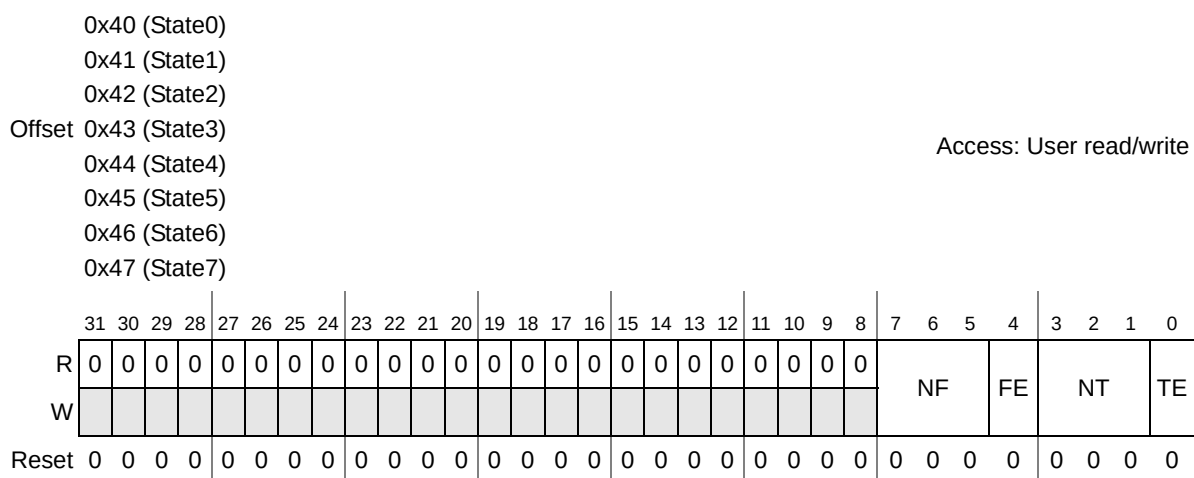


Figure 1653. SnGC register format

The SnGC register fields are described in [Table 1549](#).

Table 1549. SnGC register field descriptions

Field	Description
0 TE	GoTo Control of State Under True Condition. Defines the Goto control of a state under the true condition. If this bit is enabled, then the State moves to the next state based on bits[3:1]. If disabled, then this is the end of a sequence when the sequence comes to this state. Actions associated with states under true condition are executed. Disabling this bit ends the sequence under a true condition. 0 Disable Goto control 1 Enable Goto control
3:1 NT	Next State on True Condition. Defines the Goto next state in case of true condition from the present state. States may remain in the same state on a true condition if the same state is chosen as the next state. 000 Goto State 0 001 Goto State 1 010 Goto State 2 011 Goto State 3 100 Goto State 4 101 Goto State 5 110 Goto State 6 111 Goto State 7
4 FE	GOTO Control of State Under False Condition. Defines the Goto control of a state under the false condition. If this bit is enabled, then the state moves to the next state based on bits[7:5] . If disabled, then this is the is end of a sequence when the sequence comes to this state. Actions associated with states under false condition are executed. Disabling this bit end the sequence under a false condition. 0 Disable Goto control 1 Enable Goto control
7:5 NF	Next State on False Condition. Defines the Goto next state in case of false condition from the present state. States may remain in the same state on a false condition if the same state is chosen as the next state. 000 Goto State 0 001 Goto State 1 010 Goto State 2 011 Goto State 3 100 Goto State 4 101 Goto State 5 110 Goto State 6 111 Goto State 7
31:8	Reserved. Read returns 0.

69.5.1.4 SLU status (SS) register

The SLU status register indicates the states that have been hit/encountered in any of the active sequences. [Figure 1654](#) shows the format of the SS register.

Figure 1654. SS register format

Offset 0x48

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	IM2	IM1	IM0	EM2	EM1	EM0	SAS4			SAS3
W							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAS3		SAS2		SAS1				SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SS register fields are described in [Table 1550](#).

Table 1550. SS register field descriptions

Field	Description
0 SH0	State Hit 0. This bit is set when State0 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
1 SH1	State Hit 1. This bit is set when State1 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
2 SH2	State Hit 2. This bit is set when State2 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
3 SH3	State Hit 3. This bit is set when State3 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
4 SH4	State Hit 4. This bit is set when State4 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
5 SH5	State Hit 5. This bit is set when State5 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
6 SH6	State Hit 6. This bit is set when State6 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
7 SH7	State Hit 7. This bit is set when State7 is hit/encountered in any of the active sequences 0 State not hit 1 State hit

Table 1550. SS register field descriptions (continued)

Field	Description
10 :8 SAS1	Sequence 1 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 1 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
13 :11 SAS2	Sequence 2 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 2 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
16:14 SAS3	Sequence 3 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 3 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
19 :17 SAS4	Sequence 4 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 4 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
20 EM0	Exception Match 0. This bit indicates the status of the processor exception vector match value of CPU0. 0 No match 1 Match occurred
21 EM1	Exception Match 1. This bit indicates the status of the processor exception vector match value of CPU1. 0 No match 1 Match occurred

Table 1550. SS register field descriptions (continued)

Field	Description
22 EM2	Exception Match 2. This bit indicates the status of the processor exception vector match value of CPU2. 0 No match 1 Match occurred
23 IM0	Interrupt Match 0. This bit indicates the status of the match value on interrupt priority register of CPU0. 0 No match 1 Match occurred
24 IM1	Interrupt Match 1. This bit indicates the status of the match value on interrupt priority register of CPU1. 0 No match 1 Match occurred
25 IM2	Interrupt Match 2. This bit indicates the status of the match value on interrupt priority register of CPU2. 0 No match 1 Match occurred
31:26	Reserved. Read returns 0

69.5.1.5 SPU enable (SE) register

The SPU enable/disable register is used to enable and disable the SPU. If disabled, all inputs are masked. [Figure 1655](#) shows the format of the SE register.

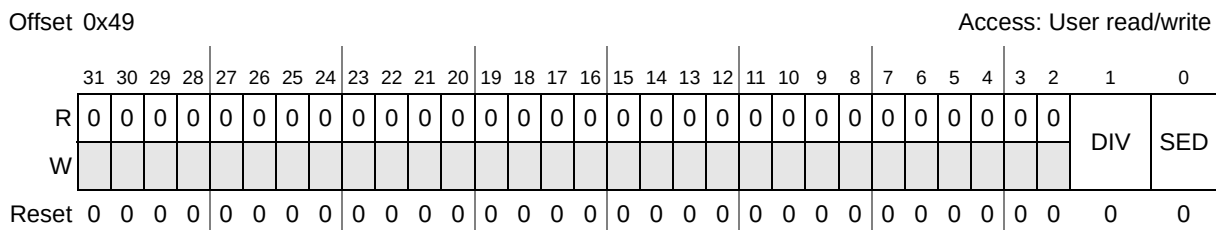


Figure 1655. SE register format

The SE register fields are described in [Table 1551](#).

Table 1551. SE register field descriptions

Field	Description
0 SED	SPU Enable/Disable. This bit should be set after programming all of the SPU configuration registers. 0 SPU is disabled 1 SPU is enabled
1 DIV	Div Option. Select Div4 and Div1 programmable options based on number of active sequences. 0 Use Div4 option, when multiple up to four sequences are active or only one sequence is active which does not require the highest resolution. 1 Use a Div1 option, when only one sequence (Sequence1) is active and highest resolution is required
31:2	Reserved

Note: *Div option mapping with active sequences must be correctly programmed by the user. No hardware checks are implemented.*

69.5.1.6 Sample and hold mechanism

SPU supports Div4/Div1 sample and state processing frequency (both are effectively the same). For the Div1 option, the sample and state processing frequency is the SPU clock frequency. Likewise, for the Div4 option, the sample and state processing frequency is 1/4th the SPU clock frequency.

The Div1 or Div4 is related to hold circuitry (among other circuits as well), but the sample circuitry is always Div1. Therefore, if Div4 is selected for hold circuitry, then four samples are taken and then latched into a hold circuitry (for the next four Div1 clocks, synchronized for all inputs).

This sample and hold mechanism is applicable for external signals only, but counters logic process differently. These external signals may be considered as selected conditions for any sequence, and for any of the states of any sequence. There is no difference based on state or sequence, as all inputs are treated the same regardless.

69.5.1.6.1 Functional limitations

If there are multiple true conditions in a Div4 period, only one condition is registered. Never are there any missed true conditions, except for extra true conditions in a Div4 period. Therefore, the user should select the resolution of SPU based on number of active sequences.

- If the signal changes faster than the sample and state frequency, then the SPU would probably only detect one event per the sample period (for example: If there are multiple true conditions in a Div4 period, only one condition is registered).
- If the signal changes faster than the sample and state frequency, then the SPU would only be able to process one event per state processing frequency.

69.5.1.6.2 Assumption

When using Div1, SPU counter operations are not available for state evaluation in the immediately next clock cycle. Therefore, special handling must be done in state evaluation during this delayed effect to assure that it does not create undesirable results. Refer to counter workaround description in [Section 69.2, SPU actions](#).

When using Div4, however, there are no delayed effects when using SPU counter operations. This is because subsequent state evaluations are done four clocks apart, instead of every clock for Div1 selection. Therefore, counter workaround description in [Section 69.2, SPU actions](#), becomes redundant with Div4 option.

69.5.1.7 SLU action unit registers

The action unit of the SPU is responsible for generating the action triggers supported by various actions. List of the actions is given in [Section 69.2, SPU actions](#). The action triggers are mapped to the events by programming the action registers. Each event can be programmed to trigger interrupts, watchpoints, trace signals, counter stop/reset/increment and many other general actions. [Table 1552](#) shows the bit mapping for all the possible actions. All possible actions are divided in three groups: client based, counter based, system related.

Table 1552. SPU action decoder

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
00	Client group	00000	CPU0	0000	No action	—	0=Stop 1=Start
		00001	CPU1	0001	DTM/gtm_gen0 trace	Valid for Cores, AHB, GTM and trace group actions.	
		00010	CPU2	0010	OTM/gtm_gen1 trace		
		00011–00111	Reserved	0011	PTM/gtm_gen2 trace		
		01000	GTM	0100	WTM/gtm_gen3 trace		
		01001–01111	Reserved	0101	PMC1/NAR_supp_trig1		
		10000	NXMC1	0110	PMC2/NAR_supp_trig2		
		10001	NXMC2	0111	PMC3/NAR_supp_trig3		
		10010–10111	Reserved	1000	PMC4/NAR_supp_trig4	Valid for GTM	
		11000	NAR	1001	GTM_dtm_tr0		
		11001–11110	Reserved	1010	GTM_dtm_tr1		
				1011	GTM_ftm_tr0		
				1100	GTM_ftm_tr1		
		11111	Trace group ⁽¹⁾	1101–1111	Reserved	Valid for CPU cores and AHB	0=Stop 1=Start
01	Counter group	00000	Counter0	0000	No action	—	Don't care
		00001	Counter1	0001	Start timer/Incr counter	Valid for all counters/timers	
		00010	Counter2	0010	Stop timer		
		...	...	0011	Reset counter		
		01111	Counter15	0100	Capture		
		10000–11111	Reserved	0101–1111	Reserved		

Table 1552. SPU action decoder (continued)

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]	
ID	Name	ID	Name	ID	Name	Comment		
10	System group	00000	System action	0000	No action	—	Don't care	
		00001–11111	Reserved	0001	Halt1_dci_evti	—		
				0010	Halt2_dci_evti	—		
				0011	Timestamp0_ahb1	Valid for the AHB clients		
				0100	Timestamp1_ahb1			
				0101	Timestamp0_ahb2			
				0110	Timestamp1_ahb2	Valid for NAR		
				0111	Evto1			—
				1000	Evto2			—
				1001	NAR_sync_trigg	Valid for NAR		
				1010	Core INTR	—		
		1011–1111	Reserved	—	—			
11	Reserved							

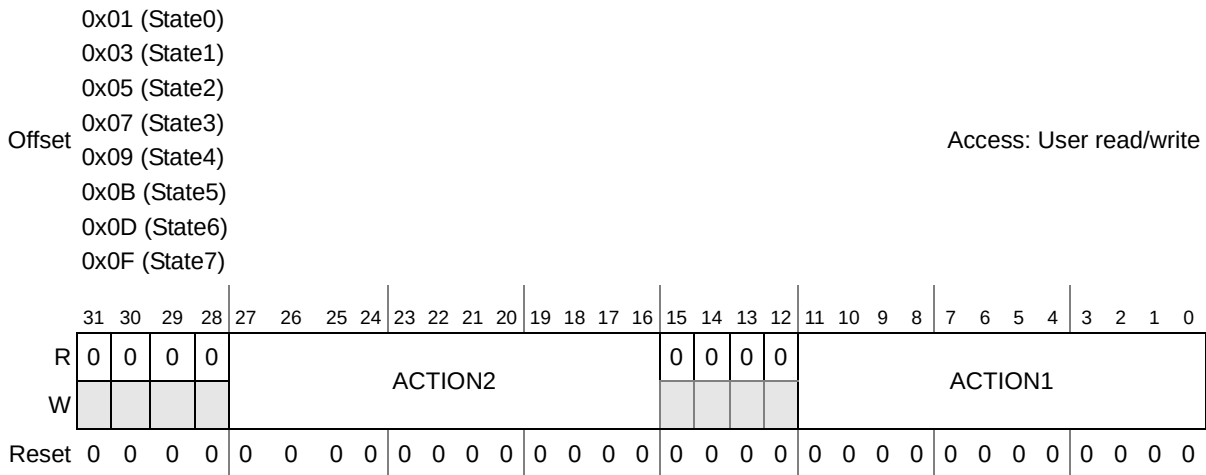
1. The valid actions for Trace group are OTM trace, PTM trace, DTM trace or WTM trace. No other actions can be selected with Trace group. The group can be defined using the Trace group configuration registers.

69.5.1.7.1 State n true action 0 (SnTA0)

The SPU can be programmed to enable up to four different actions against a True condition of every state. All the possible actions and their encoding are listed in [Table 1552](#). A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1.

[Figure 1656](#) shows the format of the SnTA0 register, where n, the state number, is equal to 0–7.

Figure 1656. SnTA0 register format



The SnTA0 register fields are described in [Table 1553](#).

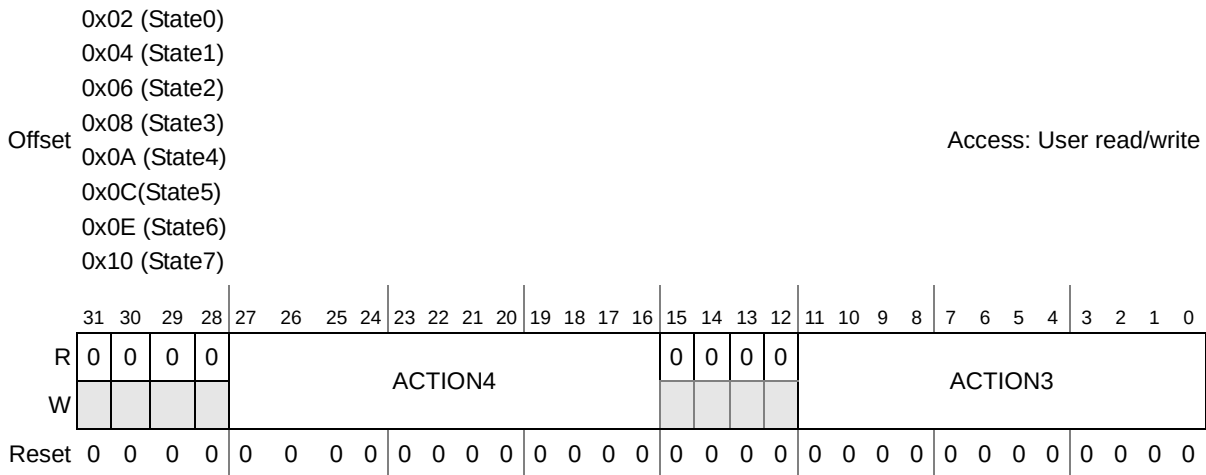
Table 1553. SnTA0 register field descriptions

Field	Description
0:11 ACTION1	Action 1 to be performed for State n true condition. The decoding of these bits is defined in Table 1552
12:15	Reserved
16:27 ACTION2	Action 2 to be performed for State n true condition. The decoding of these bits is defined in Table 1552
28:31	Reserved

69.5.1.7.2 State n true action 1 (SnTA1)

The SPU can be programmed to enable up to four different actions against a True condition of every state. All the possible actions and their encoding are listed in [Table 1552](#). A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1. [Figure 1657](#) shows the format of the SnTA1 register, where n, the state number, is equal to 0–7.

Figure 1657. SnTA1 register format



The SnTA1 register fields are described in [Table 1554](#).

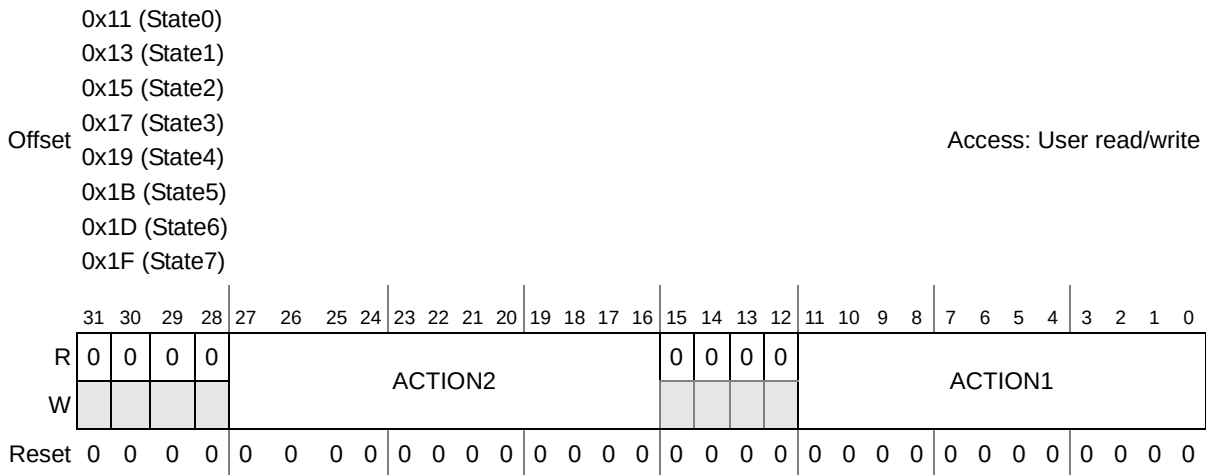
Table 1554. SnTA1 register field descriptions

Field	Description
0:11 ACTION3	Action 3 to be performed for State n true condition. The decoding of these bits is defined in Table 1552
12:15	Reserved
16:27 ACTION4	Action 4 to be performed for State n true condition. The decoding of these bits is defined in Table 1552
28:31	Reserved

69.5.1.7.3 State n false action 0 (SnFA0)

The SPU can be programmed to enable up to four different actions against a False condition of every state. All the possible actions and their encoding are listed in [Table 1552](#). A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. [Figure 1658](#) shows the format of the SnFA0 register, where n, the state number, is equal to 0–7.

Figure 1658. SnFA0 register format



The SnFA0 register fields are described in [Table 1555](#).

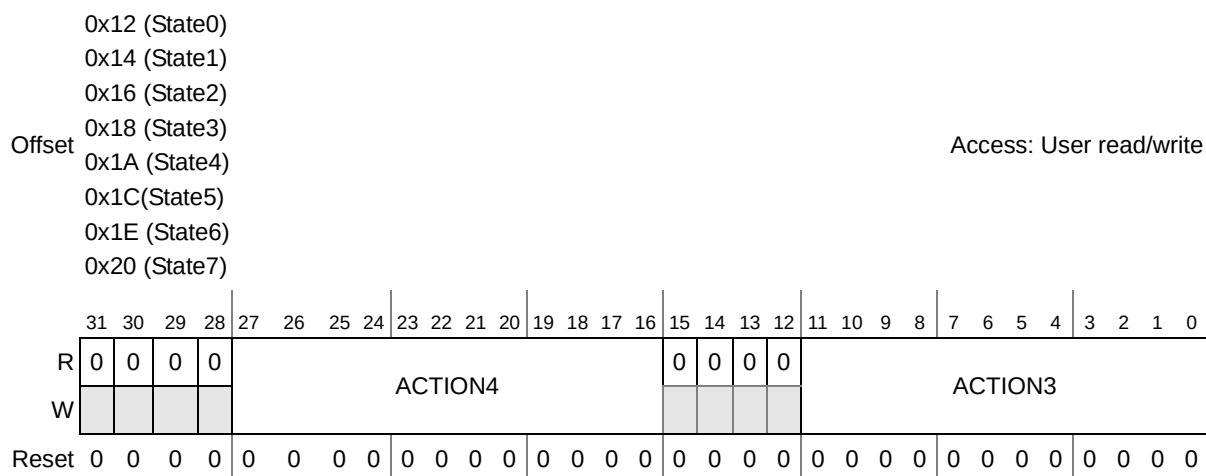
Table 1555. SnFA0 register field descriptions

Field	Description
0:11 ACTION1	Action 1 to be performed for State n false condition. The decoding of these bits is defined in Table 1552
12:15	Reserved
16:27 ACTION2	Action 2 to be performed for State n false condition. The decoding of these bits is defined in Table 1552
28:31	Reserved

69.5.1.7.4 State n false action 1 (SnFA1)

The SPU can be programmed to enable up to four different actions against a False condition of every state. All the possible actions and their encoding are listed in [Table 1552](#). A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. [Figure 1659](#) shows the format of the SnFA1 register, where n, the state number, is equal to 0–7.

Figure 1659. SnFA1 register format



The SnFA1 register fields are described in [Table 1556](#).

Table 1556. SnFA1 register field descriptions

Field	Description
0:11 ACTION3	Action 3 to be performed for State n false condition. The decoding of these bits is defined in Table 1552
12:15	Reserved
16:27 ACTION4	Action 4 to be performed for State n false condition. The decoding of these bits is defined in Table 1552
28:31	Reserved

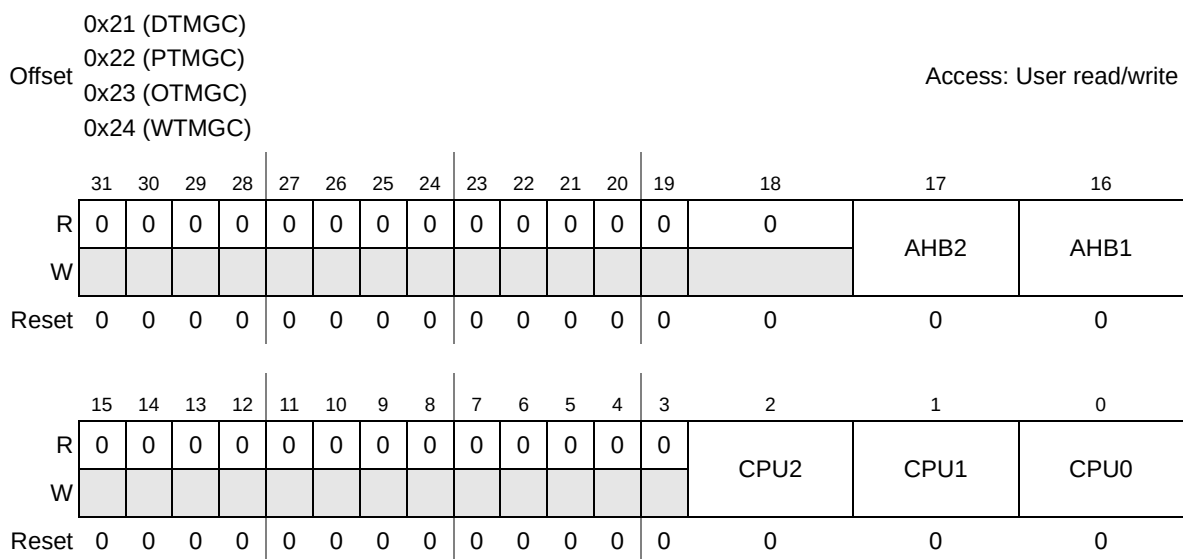
69.5.1.7.5 Trace group configuration registers (DTMGC, PTMGC, OTMGC and WTMGC)

There are four action trace groups and a configuration register for each trace group:

- DTM trace—clients are configured with the DTM trace group configuration (DTMGC) register
- PTM trace—clients are configured with the PTM trace group configuration (PTMGC) register
- OTM trace—clients are configured with the OTM trace group configuration (OTMGC) register
- WTM trace—clients are configured with the WTM trace group configuration (WTMGC) register

Each of these groups are specific to a trace. Any number of clients can be enabled for one trace using the action trace group.

[Figure 1660](#) shows the format of the trace group configuration registers.

Figure 1660. Trace group configuration register format

The trace group configuration register fields are described in [Table 1557](#).

Table 1557. Trace group configuration register field descriptions

Field	Description
0 CPU0	0Disable trace for CPU0 1Enable trace for CPU0
1 CPU1	0Disable trace for CPU1 1Enable trace for CPU1
2 CPU2	0Disable trace for CPU2 1Enable trace for CPU2
3:15	Reserved
16 AHB1	0Disable trace for AHB1 1Enable trace for AHB1
17 AHB2	0Disable trace for AHB2 1Enable trace for AHB2
18:31	Reserved

69.5.1.7.6 Interrupt status (INTS)

The status of the external interrupt request to the processor core through the JTAGM is registered in the Interrupt status (INTS) register. This status bit is cleared when the acknowledge is received from the JTAGM. [Figure 1661](#) shows the format of the INTS register.

Figure 1661. INTS register format

Offset 0x25																Access: User read/write																
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The INTS register fields are described in [Table 1558](#).

Table 1558. INTS register field descriptions

Field	Description
0 INT	1Interrupt status bit is cleared from JTAGM
31:1	Reserved

69.5.1.8 Counters/timers control n (CTRLn) registers

There are 16 configurable counters/timers in the SPU. Configuration control of these counters is defined in the counter control registers. [Figure 1662](#) shows the format of the CTRLn registers, where n, the counter number, is equal to 0–15.

Figure 1662. CTRLn register format

Offset 0x26–0x35 (CTRL0–15)																Access: User read/write							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16							
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
W																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0	0	0	0	0	0	0	0	SW_RESET		PRESCALING		0	COUNT_INCR		MODE	ENABLE
W								0									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The CTRLn register fields are described in [Table 1559](#).

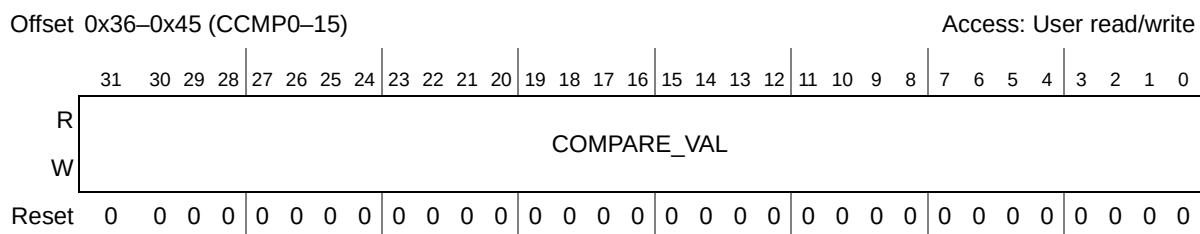
Table 1559. CCTRLn register field descriptions

Field	Description
0 ENABLE	Enable. 0 Counter disabled 1 Counter enabled
1 MODE	Mode Select. Select counter or timer operation. 0 Counter 1 Timer
2 COUNT_INCR	Count Increment. 0 Increment the counter based on the actions generated from the sequences 1 Increment the counter based on the PMC input. The mapping of the PMC inputs with the counters is shown in Figure 1674 .
3	Reserved. Read returns 0.
6:4 PRESCALING	Prescaling. Divides the timer clock by a defined value. 000 No scaling, divide by 1 001 Divide by 4 010 Divide by 16 011 Divide by 32 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 SW_RESET	Reset. Counter can be reset by software 0 Software reset is disabled 1 Software reset
8 Reserved	This bit is for internal use only and must be set to 0.
31:9	Reserved. Read returns 0.

69.5.1.9 Counters/timers compare n (CCMPn) registers

A 32-bit comparator value can be programmed for each of the counters. This value is compared with the corresponding counter/timer to generate an event for the SPU. [Figure 1663](#) shows the format of the CCMPn registers, where n, the counter number, is equal to 0–15.

Figure 1663. CCMPn register format



The CCMPn register fields are described in [Table 1560](#).

Table 1560. CCMPn register field descriptions

Field	Description
31:0 COMPARE_VAL	Compare value. This value is compared with the corresponding counter/timer value to generate an event for the SPU.

69.5.1.10 Status registers

69.5.1.10.1 Counter compare status (CCOMS)

The counter compare status register shows the status of the comparisons for all the counters. The bits in this register can be cleared by writing a 1 to them. [Figure 1664](#) shows the format of the CCOMS register. All of the CC fields in the CCOMS register have the same description as shown in [Table 1561](#).

Figure 1664. CCOMS register format

Offset 0x46												Access: User read/write				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCOMS register fields are described in [Table 1561](#), where n is the counter number.

Table 1561. CCOMS register field descriptions

Field	Description
CCn	Counter n Compare. 0 Counter status is not true 1 Counter status is true

69.5.1.10.2 Counter overflow status (COS)

Whenever a counter overflow occurs, the corresponding counter overflow status register bit is set. The bits in this register can be cleared by writing a 1 to them. [Figure 1665](#) shows the format of the COS register. All of the CO fields in the COS register have the same description as shown in [Table 1562](#).

Figure 1665. COS register format

Offset 0x47												Access: User read/write				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CO15	CO14	CO13	CO12	CO11	CO10	CO9	CO8	CO7	CO6	CO5	CO4	CO3	CO2	CO1	CO0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The COS register fields are described in [Table 1562](#), where n is the counter number.

Table 1562. COS register field descriptions

Field	Description
CO _n	Counter n Overflow. 0 Counter did not overflow 1 Counter overflow occurred

69.5.1.10.3 Counter capture status (CCAPS)

Whenever a counter value is captured, the corresponding capture status register bit for that counter is set. If the counter capture value causes a FIFO overflow, and thus the captured value is lost, the status bit is not set. The bits in this register can be cleared by writing a 1 to them. [Figure 1666](#) shows the format of the CCAPS register.

All of the CC fields in the CCAPS register have the same description as shown in [Table 1563](#).

Figure 1666. CCAPS register format

Offset 0x48												Access: User read/write				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCAPS register fields are described in [Table 1563](#), where n is the counter number.

Table 1563. CCAPS register field descriptions

Field	Description
CCn	Counter n Captured. 0 Counter not captured 1 Counter captured

69.6 Functional description

69.6.1 Input Mux unit

The SPU receives numerous inputs in the form of complex triggers from various clients to create a predefined sequence. One state logic can accommodate 16 inputs at one point of time. Therefore, an input Muxing unit becomes essential for reducing the debug and performance events to a manageable set of 16 inputs for each SLU. In addition to client input signals, there are certain performance events which are fed directly to the different counters inside the SPU.

The selection of 16 inputs for each state logic is done at two level of muxing both of which are shown in [Table 1667](#).

- Level 1 Muxing—At this level, there are 209 inputs. Out of these possible inputs, 63 inputs are selected. This selection is done using an 8:1 Mux. Therefore, sixty-four 8×1 muxes are required. The set of 63 watchpoints are passed to the second level of muxing. The selection mechanism is shown in [Figure 1667](#).
- Level 2 Muxing—At this level, 16 inputs are selected out of the 63 possible inputs for each SLU. These 63 inputs consist of multiple set of 8 watchpoints from different clients, processor exception signals, interrupt from clients, and event IN/OUT signals. In addition, certain events from counters are used as feedback in the present state logic of each state.

At level two there are 16 banks of 64:1 muxes for the selection of 16 inputs for one state logic. The inputs to these Muxes are expected to be somewhat unique. In general, care should be take to choose signal assignments that allow for key use cases to be achieved without encountering concurrency restrictions. Since there are eight such states, 16×8 banks of 64:1 mux are required. The selection mechanism is shown in [Figure 1667](#).

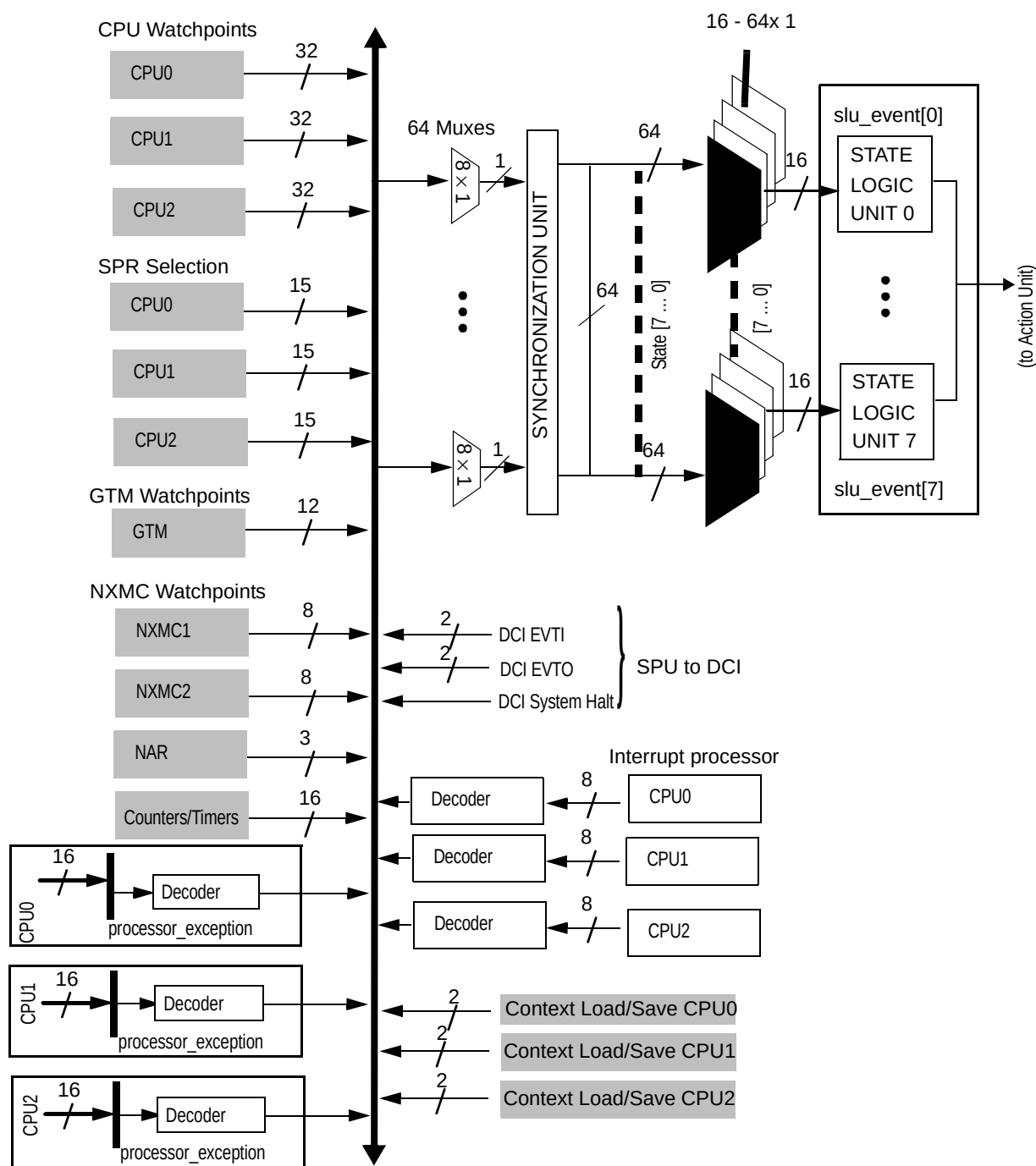


Figure 1667. Input Mux unit block diagram

69.6.2 Processor exception vector encoding

Active-high input signals from the CPU indicate the particular exception being processed when the CPU exception enable signal is asserted. These signals can be used for debug triggering mechanisms. The SPU captures the CPU exception vector whenever the CPU exception enable signal is asserted and the SPU uses the exception vector as an input

condition as shown in [Table 1564](#). The CPU exception vector values from the CPUs must be byte swapped before they are programmed to the CPU_n field of the CnPEVP (n = 0–2) registers as shown in [Table 1564](#).

Table 1564. Processor exception processing conditions

CPU exception vector	CnPEVP register CPU _n field value	Exception condition
0x0000	0x0000	Critical Input—autovectored
0x0010	0x1000	Machine check
0x0020	0x2000	Data storage
0x0030	0x3000	Instruction storage
0x0040	0x4000	External input—autovectored
0x0050	0x5000	Alignment
0x0060	0x6000	Program trap
0x0061	0x6100	Program illegal
0x0062	0x6200	Program privileged
0x0070	0x7000	Performance monitor
0x0080	0x8000	System call
0x0090	0x9000	Debug
0x00A0	0xA000	EFPU data exception
0x00B0	0xB000	EFPU round exception
0xVVVV ₃ , low two bits = 01	0xVVVV ₃ , high two bits = 01	External input—vectored
0xVVVV ₃ , low two bits = 01	0xVVVV ₃ , high two bits = 01	Critical input—vectored

69.6.3 State logic unit (SLU)

Complex triggers and system performance monitor functions are implemented in the SLU. The SPU creates debug events based upon states in a sequence. A sequence can consist of a single state, or any number of states up to 8. Each state consists of combinational logic allowing AND/OR operations on inputs from the trigger source unit. Single or multiple (up to 8) actions can be triggered by a state machine.

A logic diagram of one SLU state is shown in [Figure 1668](#).

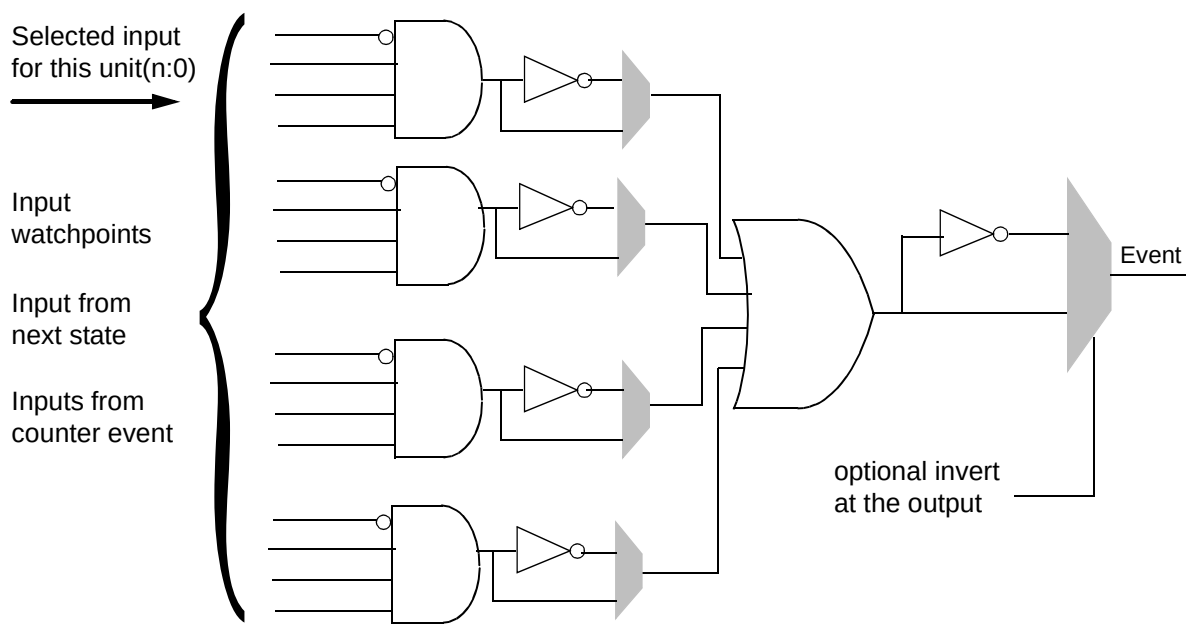


Figure 1668. SLU block diagram

Each state consists of AND gates, the outputs of which are ORed together.

- Each AND gate can have up to 4 inputs, selected from the trigger source unit by the user.
- The user has an option to invert one input signal into the AND gate.
- The OR gate has a user selectable option to invert the output.
- The output of the OR gate is used as the trigger to move to the next state in the sequence.

The SLUs are responsible for performing operations on the selected input events to produce derived events. Each SLU has a control register that manages how the various inputs are combined to produce a derived event.

An event is triggered from each state based on all the possible selected inputs for each state. This event is used to trigger an Action.

Note: With only one optional inversion at the input of each AND gate and optional inversion at the output of the OR gate, many combinations are not possible.

69.6.4 Sequence formation

Event sequencing can be accomplished with the if-then-else operations. The next event can be fed back as a necessary condition for the current event. For example, event5 can be designated as a necessary condition for event4 in case test condition fails in event5. Event5 can be in turn be designated as necessary for event6 in case test condition passes, and so on. In this way, events can be chained together to create a sequence.

To create complex triggers, a configurable state machine is implemented. This allows the user to join states together with if-then-else operations to create a sequence. The SPU supports up to four simultaneous sequences, however only eight states are supported regardless of the number of active sequences, and each state can only be used in one unique sequence.

For example, the following state configurations are possible:

- 1 active sequence, with 8 states
- 2 active sequences, each with 4 states
- 2 active sequences, the first with 3 states and the second with 5 states
- 4 active sequences, each with 2 states
- 4 active sequences, each with 1 state

Each sequence has the following options which are shown in [Figure 1669](#):

- The ability to optionally trigger one or more actions based on a true condition from any state in the sequence.
- The ability to optionally trigger one or more actions based on a false condition from any state in the sequence.
- Each state in a sequence has the ability to route to another state on a true condition from the state logic, and the ability to route to a different state based on a false condition from the state logic

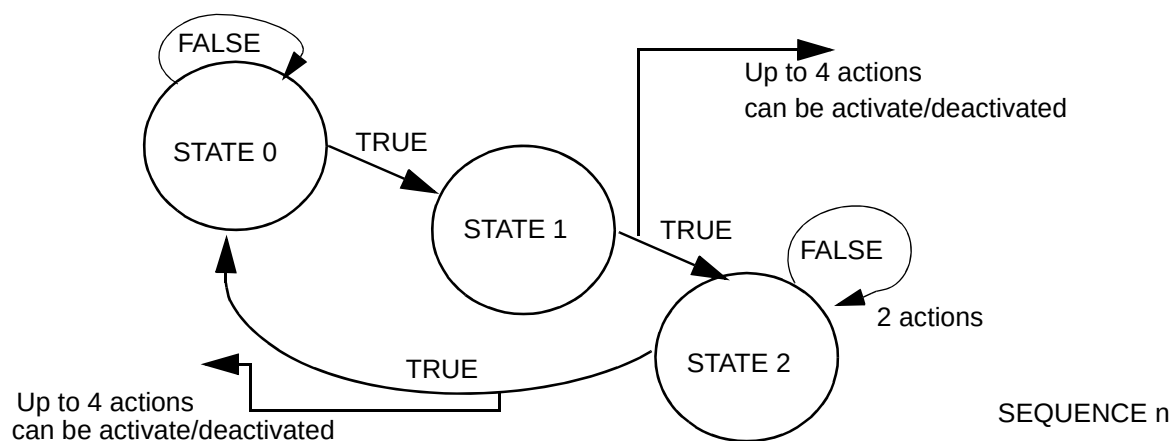


Figure 1669. Example of a simple configurable if-then-else sequence

69.6.4.1 Complex sequence example

A user can create a complex sequence using five states as shown in the state transition diagram in [Figure 1670](#).

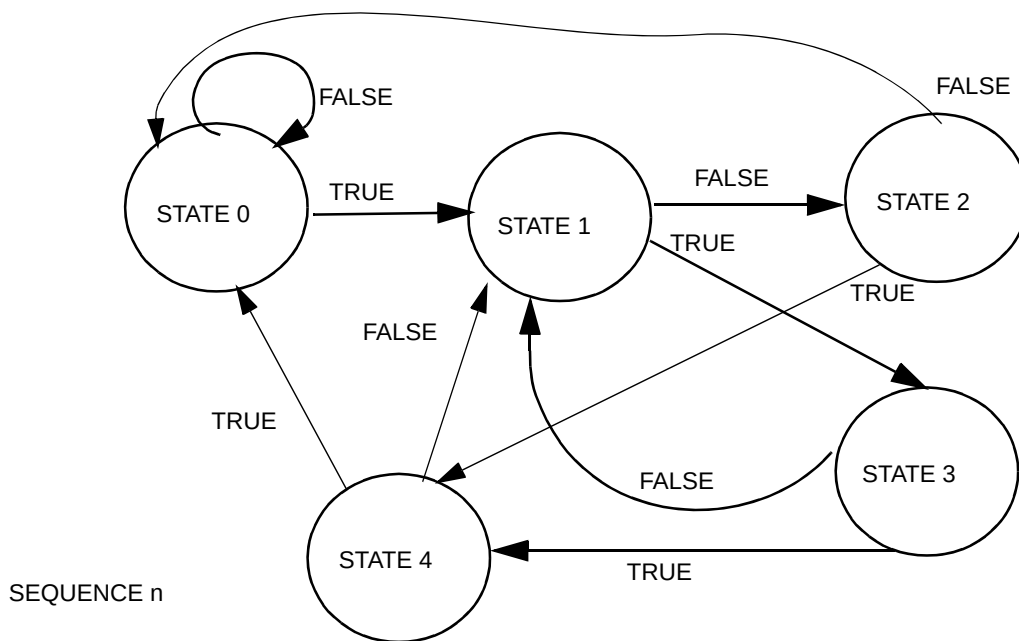


Figure 1670. Complex sequence example

69.6.4.2 Typical sequence example

For this example, the user would like to create a timestamp in the trace stream each time a function is entered, and upon the 12th time of entering the function, stop the trace signal. This use case can be implemented as follows:

- State 0 takes an instruction address compare as its input. On a match of the instruction address to a specific function, called function_x, the sequence moves to State 1. Upon this action, State0 is configured to insert a timestamp into the trace stream and also increment an SPU counter, say counter[0].
- The sequence is now in State 1. State 1 takes an input from counter[0]. Counter[0] has been pre-configured to create a match event when the counter value is equal to 12.
 - If the value of counter[0] is less than 12, then the sequence moves back to State0.
 - If the value of counter[0] is equal to 12, then the sequence completes and an action from State1 disables the trace signal.

69.6.4.3 Sequence status register description

A status register providing feedback on each sequence is provided to allow debug tools to identify the source of a debug event from the SPU. The status register consists of two field types:

- States—each state has a sticky bit that is set upon a true match of the state logic. This sticky bit is cleared upon a write to this register.
- Sequence status—each sequence has a field indicating the active state of the state machine. This is a read only field.

69.6.5 Performance counters/timers unit

The SPU counters are 32-bit counters that may be used to delay events, count distances, count latencies or count events as shown in [Figure 1671](#). Therefore, these counter are used as either counters or timers. The selection between counter or timer for each of the counters is done via configuration register. The basic counting operation is straightforward. Once enabled (default is disabled mode), whenever the selected input is driven to a logic 1, the counter increments using the platform clock.

Each of the counter/timer registers are accessible (via the JTAG interface) for the configuration of timer/counter, the compare registers and global SPU timer/counter status registers. These register are accessible via the JTAG interface even when the SPU is in disabled mode.

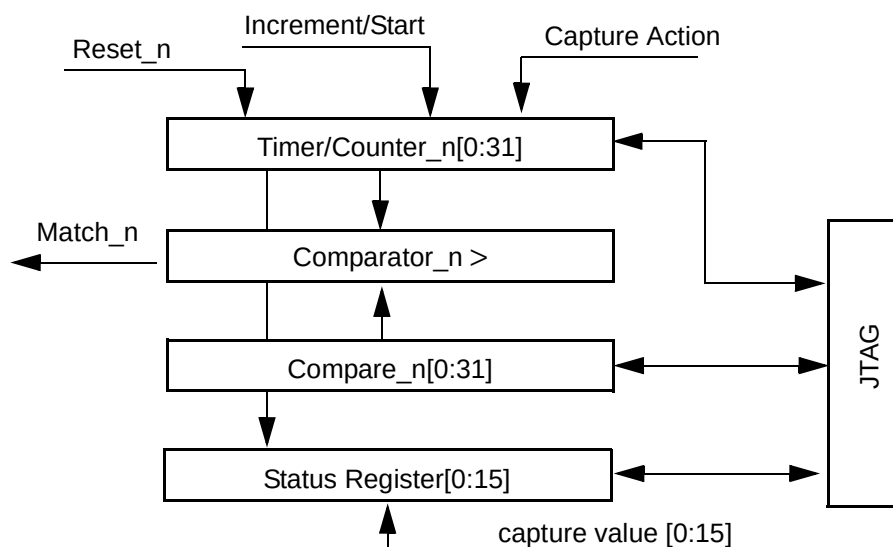


Figure 1671. Counters/timers overview

69.6.5.1 Counter events and actions

Each counter is capable of detecting one event condition, such as compare value match. The compare value match is used for triggering actions in the SPU. Each counter can be reset in response to a reset action generated from the SPU. Capture registers are available for capturing the counter state in response to a SPU trigger condition. This capture information is updated in the status register which is accessible to users via JTAG interface. These status registers are if type write 1 to clear (W1C).

There are two classes of signals, actions and events:

- **Actions**—each timer/counter has an interface from the action unit to start/increment and reset as appropriate. The start/increment signal is a single signal, with the

operation selected in the SPU configuration registers. The generation of the appropriate count/reset signal is generated in the SPU.

On an action to insert the timer/counter value into the trace stream, the value stored in the timer/counter is buffered inside the SPU and routed to the appropriate SoC block to insert the value into the trace stream one by one.

- Events—on a match event, whereby the a timer/counter value matches the value of the respective compare register, a signal is sent to the trigger source unit to indicate this.

69.6.5.2 Counter input selection at SPU level

There are two main inputs that are considered to increment the count of a counter. Each counter has the option to increment itself either from the SPU action or direct from the CPU performance measurement counters (CPU_PMC), as shown in [Figure 1672](#). These inputs include signals from the primary SPU muxes or SPU events of counters.

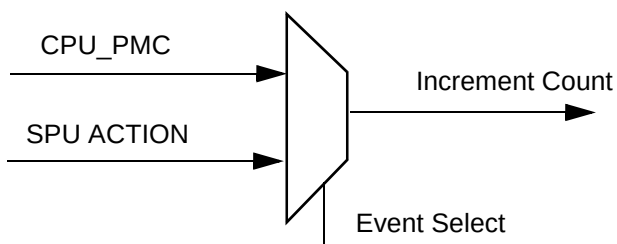


Figure 1672. Input selection for count increment

69.6.5.3 Counters increment assignment

When configured as a counter, the counter has an increment signal. This can optionally be routed from the SPU action unit, or directly from a CPU performance counter signal. The assignment of which signal is used to increment a counter is handled in the SPU, and is configurable by user. The assignment of increment signals to counters is shown in [Figure 1673](#).

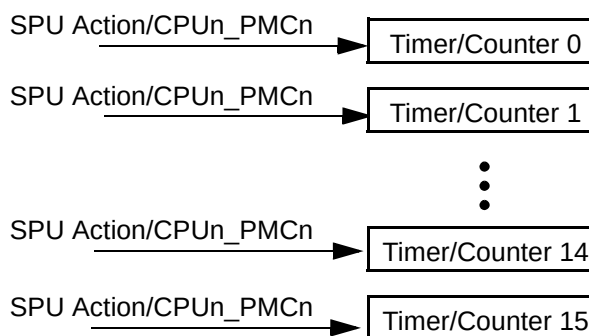


Figure 1673. SPU counters increment assignment

69.6.5.4 CPU PMC mapping to counters

Each of the CPUs in the system has four performance measurement counters (PMC). The outputs (overflow signals) from these counters (of CPU_n) are routed to the SPU debug unit. The counters inside the SPU can be configured to count the pulses from the PMCs. There is a one-to-one mapping between the PMC inputs and the counters that they trigger. Their mapping is shown in the [Figure 1674](#).

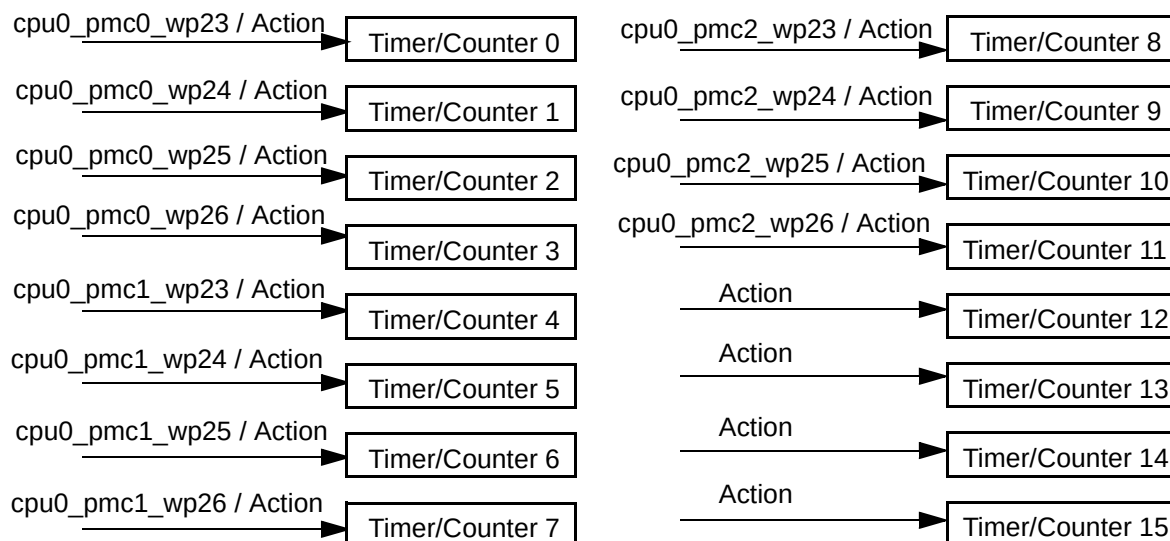


Figure 1674. CPU PMC mapping inside the SPU unit

69.6.6 Action unit

The action unit block of the SPU is responsible for the generating the action triggers supported by various actions. A state has true and false (if-else-then) condition. Four simultaneous actions can be generated under true/false condition of a particular state. The list of the actions is given in the [Section 69.2, SPU actions](#). The actions triggers are mapped to the events by programming the set of SPU group action control for events registers. Each event can be programmed to trigger interrupts, timestamp watch points, trace signals, PMCs, start/reset/increment counters and many other general actions.

69.6.7 Optional trace group

User defined trace group can be defined through the configuration register. Configuration registers registers, defined in [Table 1557](#), are available where the user can group the different traces from same or different clients.

70 JTAG Master (JTAGM)

70.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

70.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface or from the LFAST through the parallel interface. The JTAGM has the capability to differentiate between data from software and LFAST. It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI and transfers this data to the LFAST through another parallel interface. The JTAGM has the following registers:

- Configuration register
- Status register
- Four data output registers
- RxCRC receive register
- Two data input registers

All these registers, described in detail in [Section 70.4, Memory map and register definition](#), are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

[Figure 1675](#) shows the block diagram of the JTAGM.

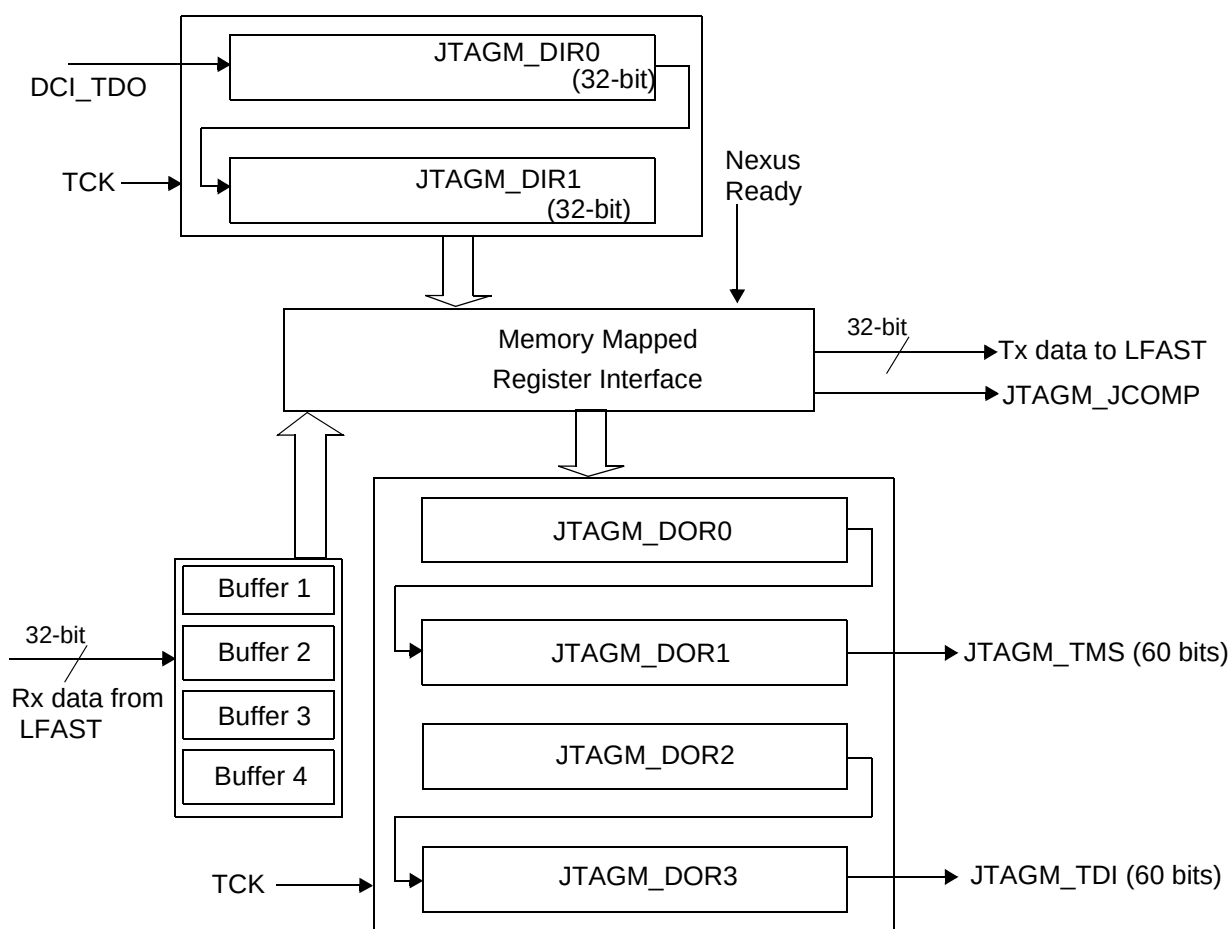


Figure 1675. JTAGM block diagram

70.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG
- Receives/provides JTAG data from/to the LFAST

70.2 Functional description

70.2.1 Module architecture

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module

interface. This concept allows complete flexible generation on TMS and TDI data and capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

Figure 1676 shows the flow diagram of the LFAST to DCI connection.

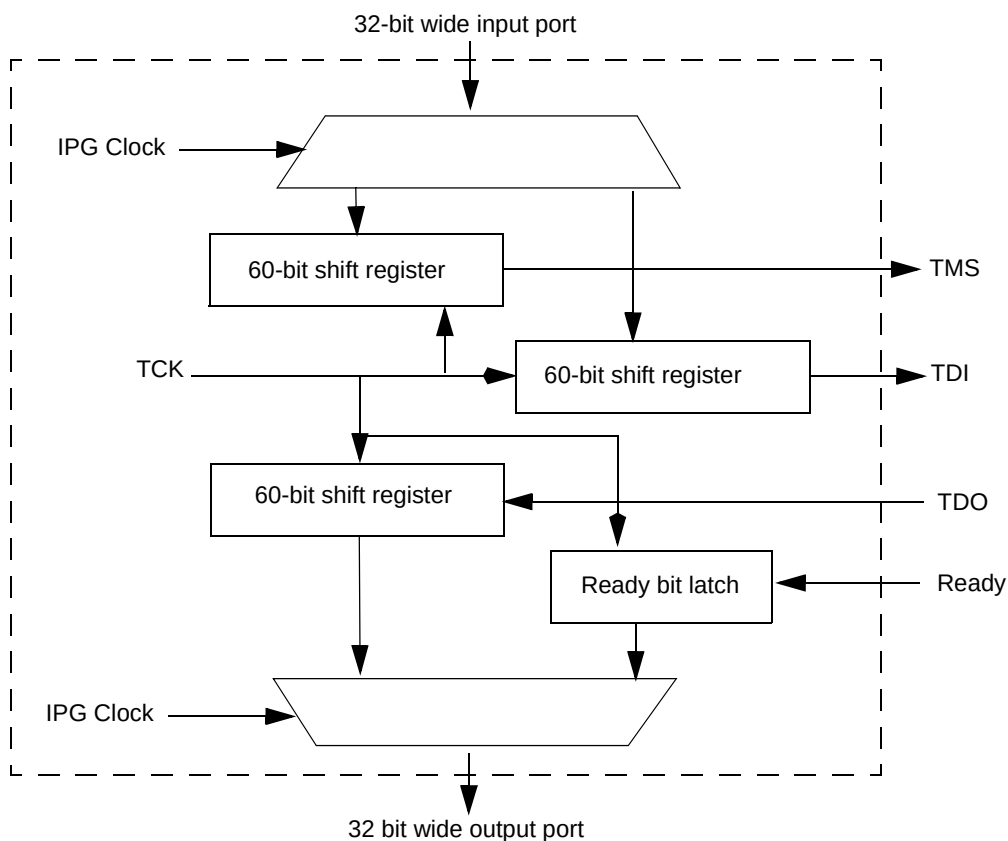


Figure 1676. LFAST-to-DCI flow diagram

70.2.2 JTAGM JCOMP usage in LFAST

JTAGM is used to drive JTAG sequences in software mode and in LFAST mode. The JTAGM_MCR[jtagm_JCOMP] bit is provided to reset all TAP controllers by generating a JCOMP signal to JTAGC.

The guidelines to be considered for the usage of JTAM_MCR[jtagm_JCOMP]:

- Do not clear the JTAM_MCR[jtagm_JCOMP] bit while in LFAST mode as this operation is not guaranteed
- If for some reasons during the LFAST mode there is a loss of communication, one of the following should be done (instead of clearing the jtagm_JCOMP bit):
 - Generate the escape sequence and then, once the device is back in JTAG mode, generate a test-logic-reset state via the TCK and TMS signals. This resets all TAP controllers but does not affect the client debug configuration
 - Generate the escape sequence and then, once the device is back in JTAG mode, force the JCOMP pin low. This resets all TAP controllers as well as the client debug configuration

Note: The escape sequence is generated by forcing LFAST LVDS ports RX+ and RX- to zero.

70.2.3 JTAM data error detection

To improve the integrity of the JTAG operation, data passed to and from the JTAM includes an 8-bit CRC only in LFAST mode. The data sent to the JTAM consists of 60 bits of TMS data, 60 bits of TDI and an 8-bit CRC. The JTAM recalculates the 8-bit CRC for the first 120 bits of data and compares the result with the received CRC. If the two match, the data is assumed to be valid and the JTAM converts this data into a valid JTAG frame.

If the CRCs do not match, the module sets the CRC error status bit (CRC_err) in the JTAM status register (JTAM_SR) and sends a standard 96-bit response frame to the LFAST channel B, with all 60 data bits set to 0. The CRC error status bit is automatically cleared as soon as the response frame has been sent to the LFAST.

Data sent from the JTAM to the LFAST also includes an 8-bit CRC. The data sent from the JTAM is in the form of a 96-bit frame consisting of 60 bits of TDO data + 4 bits of zero-padding + 32 bits of the status register. The 8-bit CRC value is calculated for the first 88 bits of the 96-bit frame and inserted into the least significant 8 bits of the status register; which is also the last 8 bits of the 96-bit frame. The tool is responsible for checking the CRC of the received data and taking corrective action.

70.2.4 JTAM message tagging

The JTAM, in conjunction with the LFAST, is pipelined to allow several JTAM messages to be sent from the tool before any response is received back. This allows better utilization of the LFAST bandwidth. To enable the tool to associate individual messages with the correct response, the messages are tagged. The tool sends data messages on LFAST channels E, F, G and H in sequence. The JTAM module returns responses on the same channel they were received. For example, if the tool sends a data command on channel E, JTAM returns the response on channel E; if the tool sends a data command on channel F, the JTAM returns the response on channel F; and so on. This way, the tool can ensure a response was generated by a specific command.

Configuration commands to write/read the configuration and status registers within the JTAM are sent on channel A and response data is also sent back on channel A.

70.2.5 JTAM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available. In conventional JTAG mode, the status of this ready bit is also provided back to

the tool via a dedicated pin to allow the tool to know data is ready and the JTAG read register can be accessed. With the LFAST interface providing pipelined data to the JTAGM, this technique is not available. Two interlocked mechanisms are provided within the JTAGM that use this ready bit to allow optimized data reads without ready bit errors.

70.2.5.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

70.2.5.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

70.2.6 $\overline{\text{EVTO}}$ and $\overline{\text{EVTI}}$ signals

The standard Nexus EVTI (Event In) and EVTO (Event Out) signals are optionally present on dedicated package pins and are used by an external tool in conjunction with the JTAG interface to pass information, events and triggers between the tool and the MCU.

The JTAGM as well as providing a higher speed alternative to the standard JTAG interface, also allows control and status of the EVTI and EVTO signals to be embedded into the serial protocol. The EVTI0 and EVTI1 signals can be asserted by writing control bits in the Module Configuration Register (JTAGM_MCR). The JTAGM can be configured by bits in the Module Configuration Register (JTAGM_MCR) to respond to a rising/falling/both edge transition of either EVTO0 or EVTO1. The response is either an unsolicited serial message to the tool when in LFAST mode (DTM = 0); or a CPU interrupt when in software mode (DTM = 1).

The unsolicited LFAST message is sent on LFAST channel B to allow the tool to distinguish the message from normal JTAG response data (channels E, F, G & H) and Config Command Responses (channel A). The CPU interrupt is gated by an Interrupt enable bit, in MCR and reported by Interrupt Status flags in the Status Register (JTAGM_SR). The status bits stay set until cleared by writing to the clear bits in the JTAG_SR.

Asserting the evti0_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. Asserting the evti1_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. The DTM bit in the JTAGM_MCR register can put the DCI in LFAST or SW mode.

70.2.7 JTAGM configuration and status monitoring

The JTAGM module requires some level of configuration and status monitoring. The JTAGM includes a 32-bit configuration register (JTAGM_MCR) and a 32-bit status register (JTAGM_SR, which includes the ready bit mentioned in [Section 70.2.5.1, Status register](#)

ready bit). The configuration register is accessed via the parallel input port by a 128-bit payload arriving on channel A. The 128-bit payload is made from the following:

- 32-bit mask of which bits within the configuration register should be written +
- 32-bit value to be written to the configuration register +
- 24-bit mask of which bits within the status register should be written +
- 8 bits of zero-padding +
- 24-bits of data to be written to the status register +
- 8-bit CRC (The least significant 8 bits of the status register cannot be written because they contain automatically generated CRC data).

The masked data is transferred to the 32-bit configuration register and 32-bit status register. The JTAGM module responds to this configuration/status write by outputting a 96-bit payload to the parallel output port to channel number A. The 96-bit payload is made up of the 32-bit value of the configuration register plus 32 bits of zero-padding plus 32-bit value of the status register (including the least significant 8 bits which are an 8-bit CRC calculated on the previous 88 bits).

While the current messages are being transmitted from the second buffering (memory mapped registers), the first buffers are continuously filled up. The status register also includes status bits provided by the LFAST module and the DCI module. The JTAGM requests data from the LFAST only when at least one internal buffer is free.

70.2.8 JTAGM to DCI serial interface

Figure 1677 shows the data transfer timings between JTAGM and DCI.

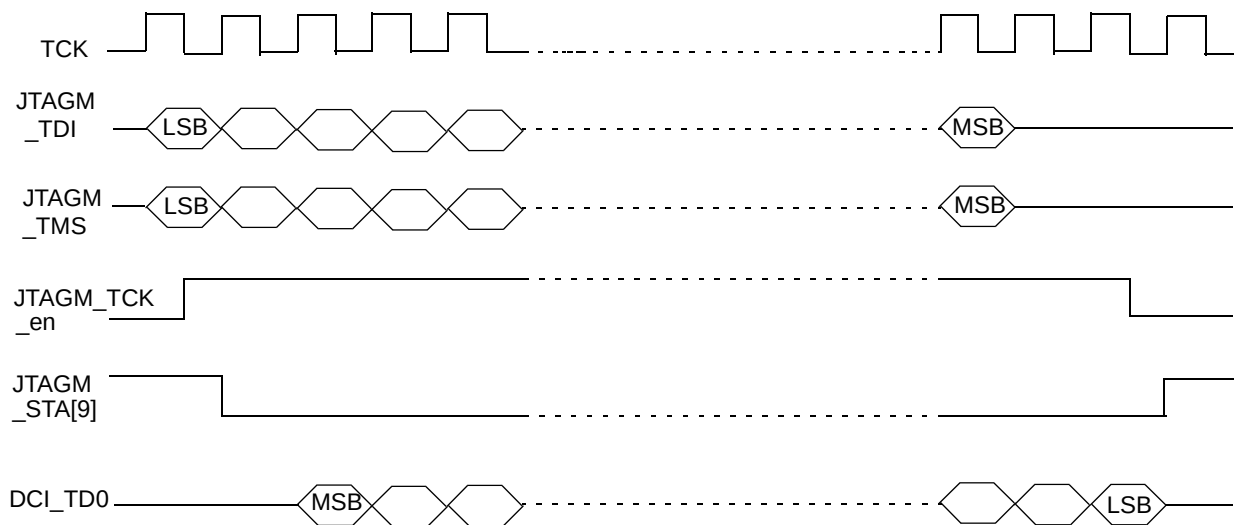


Figure 1677. JTAGM to DCI serial interface diagram

70.2.9 JTAGM data handling and CRC calculation in LFAST mode

Figure 1678 shows the bit ordering when a data frame is transmitted from LFAST to JTAGM for subsequent transmission to DCI.

bit 127 to bit 68				bit 67 to bit 8				7	6	5	4	3	2	1	0
TMS (60 bits)				TDI (60 bits)				CRC (8 bits)							
bit 0 to bit 59				bit 0 to bit 59											
Last TMS bit transmitted by JTAGM		First TMS bit transmitted by JTAGM		Last TDI bit transmitted by JTAGM		First TDI bit transmitted by JTAGM									

Figure 1678. Bit ordering for LFAST to JTAGM transfer

[Figure 1679](#) shows the bit ordering when a data frame is transmitted from JTAGM to LFAST consisting of TDO data received from DCI.

bit 95 to bit 36			bit 35 to bit 32			bit 31 to bit 8			7	6	5	4	3	2	1	0
TDO (60 bits)			0 pad (4 bits)			JTAGM_SR (24 bits)			CRC (8 bits)							
bit 0 to bit 59			bit 0 to bit 59			bit 31 to bit 8										
First TDO bit received by JTAGM		Last TDO bit received by JTAGM														

Figure 1679. Bit ordering for JTAGM to LFAST transfer

The CRC calculation is described in following example.

The data transferred from LFAST to JTAGM (in four frames) is as follows:

- D0—F7B3_D591
- D1—E6A2_C486
- D2—A2C4_80F7
- D3—B3D5_9183

These are supplied to CRC block starting from D0 to D3. The CRC calculation begins on D0 with the MSB (left bit) going first. D3 is supplied as B3D5_9100 (after masking CRC).

JTAGM extracts TMS and TDI information from the data received as follows:

- TMS = F7B3_D591_E6A2_C48
- TDI = 6A2C_480F_7B3D_591

JTAGM transmits the data to DCI starting from LSB (right bit going first). Corresponding to this TMS and TDI transmission, JTAGM receives TDO data from DCI and stores it as first bit to most significant location in TDO frame (shown in [Figure 1679](#)).

70.2.10 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

70.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

70.4 Memory map and register definition

[Table 1565](#) contains the memory map of the JTAGM registers.

Table 1565. Memory map

Offset	Register	Location
0x00	Module Configuration Register (JTAGM_MCR)	on page 3038
0x04	Status Register (JTAGM_SR)	on page 3040
0x08	Data Out Register 0 (JTAGM_DOR0)	on page 3043
0x0C	Data Out Register 1 (JTAGM_DOR1)	on page 3043
0x10	Data Out Register 2 (JTAGM_DOR2)	on page 3044
0x14	Data Out Register 3 (JTAGM_DOR3)	on page 3044
0x18	Receive CRC Register (JTAGM_RxCRC)	on page 3045
0x1C	Data Input Register 0 (JTAGM_DIR0)	on page 3045
0x20	Data Input Register 1 (JTAGM_DIR1)	on page 3046

70.4.1 Register descriptions

70.4.1.1 Module configuration register (JTAGM_MCR)

This register contains the status bits for the current transfer. [Figure 1680](#) shows the format of the JTAGM_MCR.

Figure 1680. JTAGM_MCR register

Offset 0x00																Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	SWRESET				evto0_sense		evto1_sense		evto_I E	evti0_assert	evti1_assert	0	0	0	0	0	0														
W																															
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	0	0	inter_jtag_frame_timer										0	SIE		IIE	TCKSEL		jtag_JCOMP	DTM
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0				

The JTAGM_MCR register is described in [Table 1566](#).

Table 1566. JTAGM_MCR register field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit. Writing a 0 has no effect. <i>Note: After being written to "1", this bit is auto cleared within 3 TCK cycles.</i>
1:2 evto0_sense	Determines JTAGM response to activity on $\overline{\text{EVTO0}}$ 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO0}}$ edges (default setting) 01 evto0_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO0}}$ 10 evto0_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO0}}$ 11 evto0_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO0}}$
3:4 evto1_sense	Determines JTAGM response to activity on $\overline{\text{EVTO1}}$ line 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO1}}$ edges (default setting) 01 evto1_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO1}}$ 10 evto1_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO1}}$ 11 evto1_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO1}}$
5 evto_IE	Enables $\overline{\text{EVTO}}$ triggered JTAGM interrupt 0 No JTAGM interrupt is generated in response to any $\overline{\text{EVTO0}}$ or $\overline{\text{EVTO1}}$ activity 1 JTAGM interrupt is generated when evto0_edge or evto1_edge are set
6 evti0_assert	EVTI0 Assert. 0 Set jtagm_evti0 low/asserted 1 Set jtagm_evti0 high/de-asserted
7 evti1_assert	EVTI1 Assert. 0 Set jtagm_evti1 low/asserted 1 Set jtagm_evti1 high/de-asserted <i>Note: In the SW mode (when DTM = 1), regardless of the state of "evti1_assert" bit the internal "evti" signal remains de-asserted. Though this bit can be written, yet writing to this bit has no effect.</i>
8:17	Reserved
18:23 inter_jtag_frame_timer	TCK delay. 0000000 TCK delay 0000011 TCK delay ... 11111163 TCK delay
24	Reserved
25 SIE	SPU Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer

Table 1566. JTAGM_MCR register field descriptions (continued)

Field	Description
27:29 TCKSEL	TCK clock frequency selection. 000 Not to be used 001 TCK is system clock ÷ 2 010 TCK is system clock ÷ 3 ... 111 TCK is system clock ÷ 8 <i>Note: When using the JTAGM, this bit should always be programmed with any value other than "000".</i>
30 jtagm_JCOMP	JTAG reset. 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data for JTAG transferred by LFAST 1 Data for JTAG transferred by software

70.4.1.2 Status register (JTAGM_SR)

70.4.1.2.1 LFAST mode

Figure 1681 shows the format of the JTAGM_SR in LFAST mode.

Figure 1681. JTAGM_SR register in LFAST mode

Offset 0x04																Access: User read/write															
	0		1	2		3		4		5		6		7		8	9	10		11		12	13	14		15					
R	overrun		0	lfast_status												dci_status															
W																															
Reset	0		0	0		0		0		0		0		0		0	0	0		0		0	0	0		0					

	16		17	18		19		20		21		22		23		24	25	26		27		28	29	30		31	
R	evto0_edge		evto1_edge	SPU_INT_CLR		SPU_INT		CRC_err		Nexus_err		Idle		NR		CRC											
W																											
Reset	0		0	0		0		0		0		1		0		0	0	0		0		0	0	0		0	

70.4.1.2.2 SW mode

Figure 1682 shows the format of the JTAGM_SR in SW mode.

Figure 1682. JTAGM_SR register in SW mode

Offset 0x04									Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15								
R	0	0	lfast_status						dci_status															
W																								
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0								

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_ edge	evto1_ edge	SPU_ INT_CLR	SPU_INT	CRC_err	Nexus_err	Idle	NR			0	0	0	0	0	0
W										evto0_ clr	evto1_ clr					
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

The JTAGM_SR register is described in Table 1567.

Table 1567. JTAGM_SR register field descriptions

Field	Description
0 overrun	Set by hardware when the JTAGM overwrites the first Tx data with the second Tx data. Writing 1 clears this bit. This bit is set only in LFAST mode.
1	Reserved
2:7 lfast_status	Status information from the LFAST to be used by software. The composition of these bits are as follows: bit 2 Indicates either data frame or ping frame was transmitted successfully bit 3 Indicates Tx data Interface not enabled and a frame is ready to be transmitted bit 4 Indicates Rx data FIFO overflow bit 5 Indicates reception of frame with invalid frame payload size bit 6 Indicates reception of frame with invalid ICLC code bit 7 Indicates reception of illegal LCT frame
8:15 dci_status	Status information from the DCI. The composition of these bits are as follows: bit 8 Reserved to 0 bit 9 DCI LVDS port enable bit 10 DCI JTAG port enable bit 11 DCI LVDS safe mode bit 12 DCI JTAG safe mode bit 13 DCI LVDS escape mode bit 14 DCI enable LFAST bit 15 DCI tool present , set when a debug tool is connected

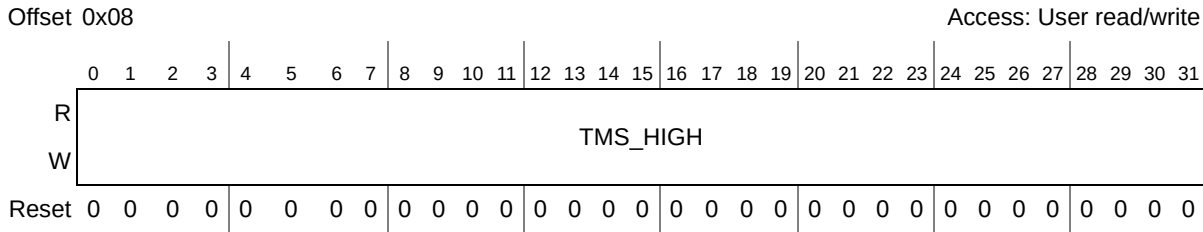
Table 1567. JTAGM_SR register field descriptions (continued)

Field	Description
16 evto0_edge	In LFAST mode (DTM = 0) a channel B message is sent on LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit 0 No activity sensed on $\overline{\text{EVTO0}}$ 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO0}}$
17 evto1_edge	In LFAST mode (DTM = 0) a channel B message is sent to the LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto1_clr bit 0 No activity sensed on $\overline{\text{EVTO1}}$ 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO1}}$
18 SPU_INT_CLR	Write 1 clears the SPU_INT This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit.
20 CRC_err	CRC error bit. Set when the calculated CRC does not match the last received CRC. This bit is cleared when the error packet is sent to the LFAST on channel B.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to 1 when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read <i>Note: In LFAST mode (when DTM = 0), this bit has no effect and acts as a "don't care". This bit should not be used by the application in the LFAST mode</i>
24:31 CRC	Calculated CRC. CRC calculated on the information received. LFAST Mode: Bit 24:31 reads the CRC. Write has no effect. SW Mode: – Bit 24:31 reads as 0x00. Writing a 0 to any bit have no effect. – Writing 1 on bit 24 clears evto0_edge – Writing 1 on bit 25 clears evto1_edge

70.4.1.3 Data out register 0 (JTAGM_DOR0)

[Figure 1683](#) shows the format of the JTAGM_DOR0.

Figure 1683. JTAGM_DOR0 register



The JTAGM_DOR0 register is described in [Table 1568](#).

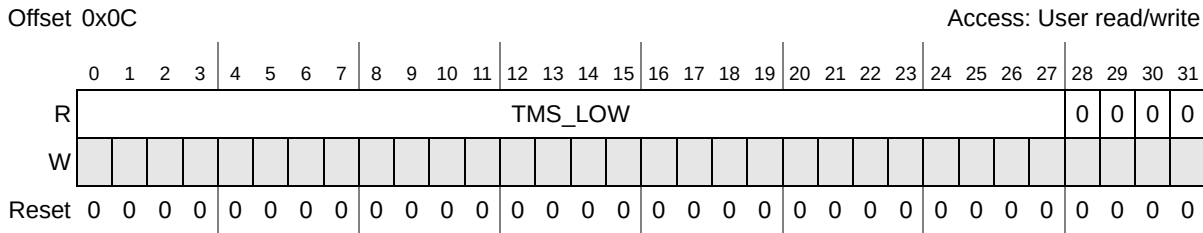
Table 1568. JTAGM_DOR0 register field descriptions

Field	Description
0:31 TMS_HIGH	Higher word of data for TMS, bits 59–28. Writable by software only when DTM = 1

70.4.1.4 Data out register 1 (JTAGM_DOR1)

[Figure 1684](#) shows the format of the JTAGM_DOR1.

Figure 1684. JTAGM_DOR1 register



The JTAGM_DOR1 register is described in [Table 1569](#).

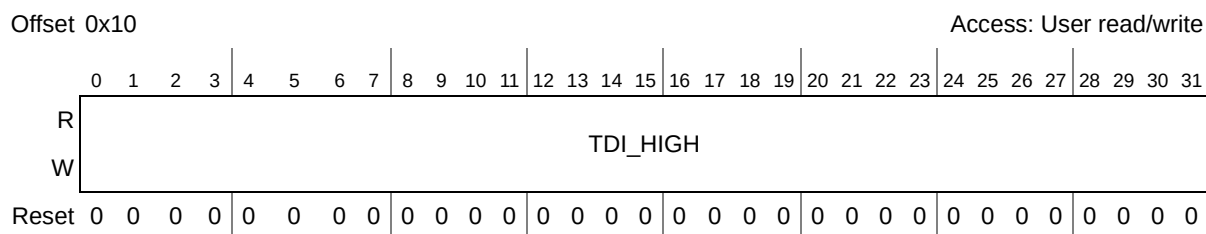
Table 1569. JTAGM_DOR1 register field descriptions

Field	Description
0:27 TMS_LOW	Lower word of data for TMS, bits 27–0. Writable by software only when DTM = 1. TMS_LOW[0] is shifted out first
28:31	Reserved

70.4.1.5 Data out register 2 (JTAGM_DOR2)

Figure 1685 shows the format of the JTAGM DOR2.

Figure 1685. JTAGM_DOR2 register



The JTAGM DOR2 register is described in [Table 1570](#).

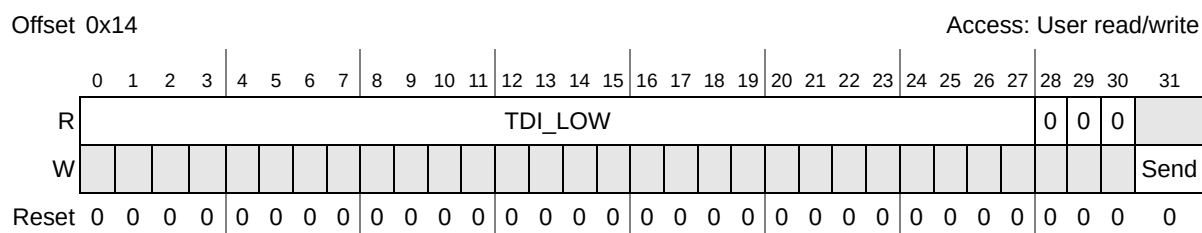
Table 1570. JTAGM_DOR2 register field descriptions

Field	Description
0:31 TDI_HIGH	Higher word of data for TDI, bits 59–28. Writable by software only when DTM = 1

70.4.1.6 Data out register 3 (JTAGM_DOR3)

Figure 1686 shows the format of the JTAGM DOR3.

Figure 1686. JTAGM_DOR3 register



The JTAGM DOR3 register is described in [Table 1571](#).

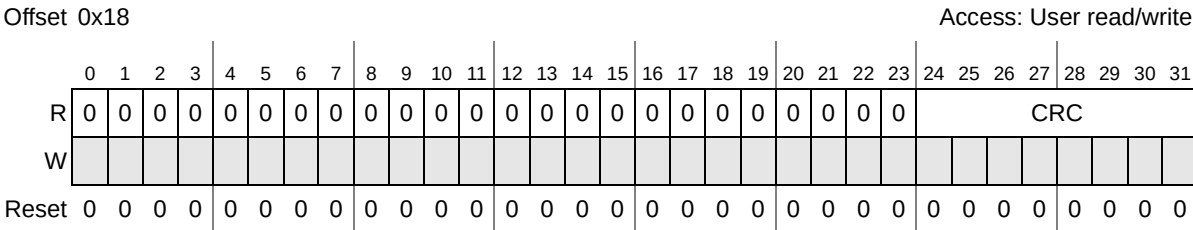
Table 1571. JTAGM_DOR3 register field descriptions

Field	Description
0:27 TDI_LOW	Lower word of data for TDI, bits 27–0. Writable by software only when DTM = 1. TDI_LOW[0] is shifted out first
28:30	Reserved
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

70.4.1.7 Receive CRC register (JTAGM_RxCRC)

[Figure 1687](#) shows the format of the JTAGM_RxCRC register.

Figure 1687. JTAGM_RxCRC register



The JTAGM_RxCRC register is described in [Table 1572](#).

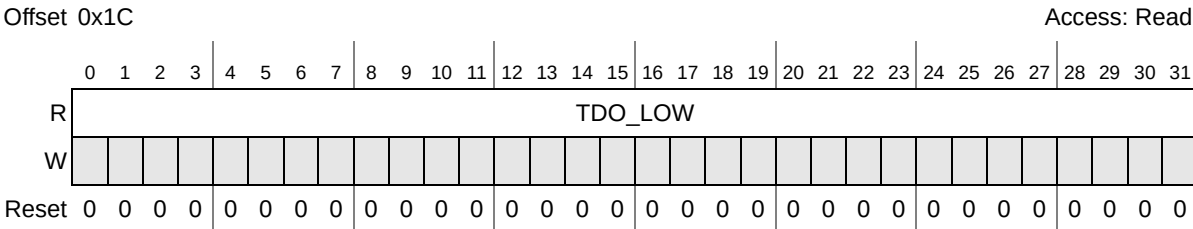
Table 1572. JTAGM_RxCRC register field descriptions

Field	Description
0:23	Reserved
24:31 CRC	Received CRC. Refers to the CRC of the received LFAST message.

70.4.1.8 Data input register 0 (JTAGM_DIR0)

[Figure 1688](#) shows the format of the JTAGM_DIR0.

Figure 1688. JTAGM_DIR0 register



The JTAGM_DIR0 register is described in [Table 1573](#).

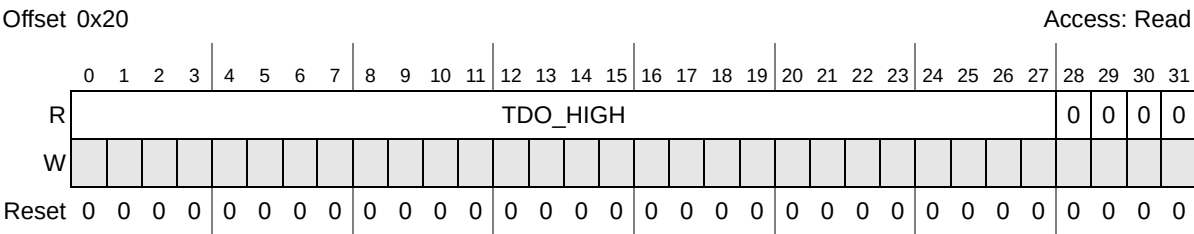
Table 1573. JTAGM_DIR0 register field descriptions

Field	Description
0:31 TDO_LOW	Lower word of data received on TDO, bits 0–31

70.4.1.9 Data input register 1 (JTAGM_DIR1)

Figure 1689 shows the format of the JTAGM_DIR1.

Figure 1689. JJTAGM_DIR1 register



The JTAGM_DIR1 register is described in Table 1574.

Table 1574. JTAGM_DIR1 register field descriptions

Field	Description
0:27 TDO_HIGH	Higher word of data received on TDO, bits 32–59
28:31	Reserved

71 LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed Debug

71.1 Introduction

This chapter describes the specifications of the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in slave only mode enabling high speed debug communications through the JTAG port.

71.2 Block diagram

Figure 1690 depicts LFAST interaction with other modules on the device.

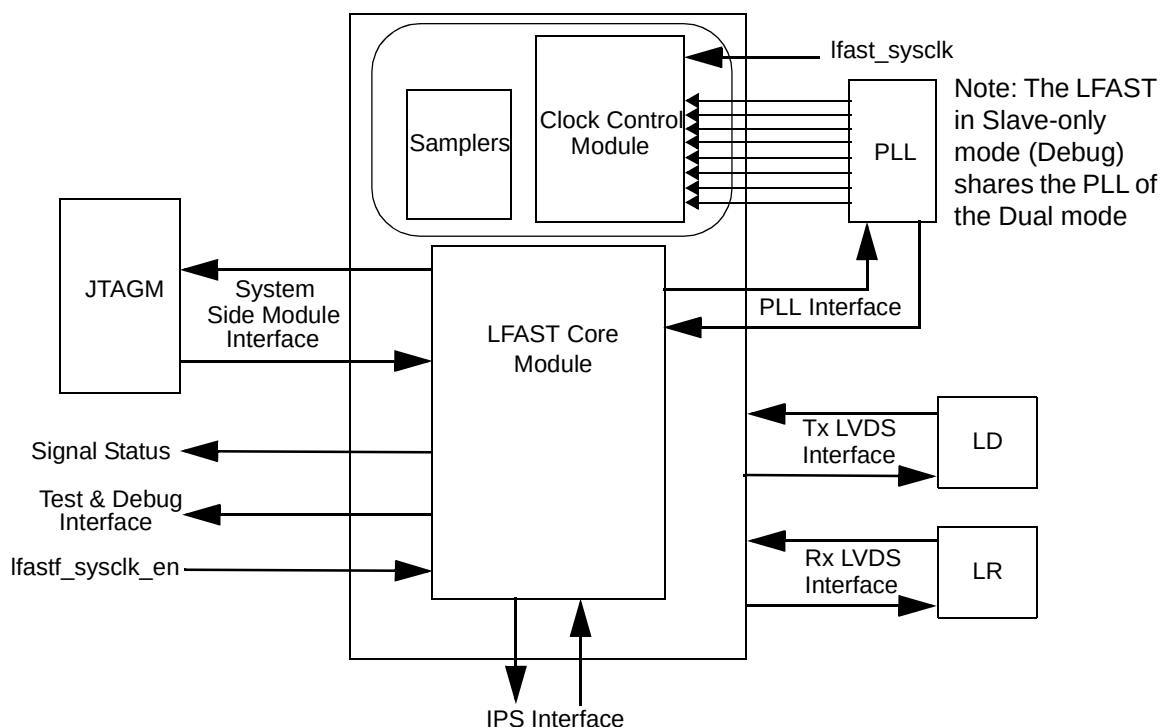


Figure 1690. LFAST block diagram

71.3 External signals

LFAST is a six pin interface with the following signals after enabling high speed debug mode:

- lfast_sysclk_en
 - Interface enable signal in LFAST Slave Only mode
- lfast_sysclk
 - Reference clock of the LFAST master and slave
- txdatap/txdatan
 - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
 - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface with maximum data rate of 320 Mbps.

71.3.1 LFAST operating data rates

[Table 1575](#) shows the operating modes for the LFAST interface.

Table 1575. Operating modes for LFAST interface

Interface	Tx/Rx Data Rate
LFAST Transmit	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps
LFAST Receive	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps ($\text{lfast_sysclk} \div 4$ or $\text{lfast_sysclk} \div 2$) is referred to as low and data rates 312 Mbps and 320 Mbps is referred to as high data rates.

71.4 LFAST frame structure

LFAST frames are transmitted and received across a serial interface using a packeted asynchronous communication format.

A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length. A sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See [Figure 1691](#) for details of the LFAST frame.

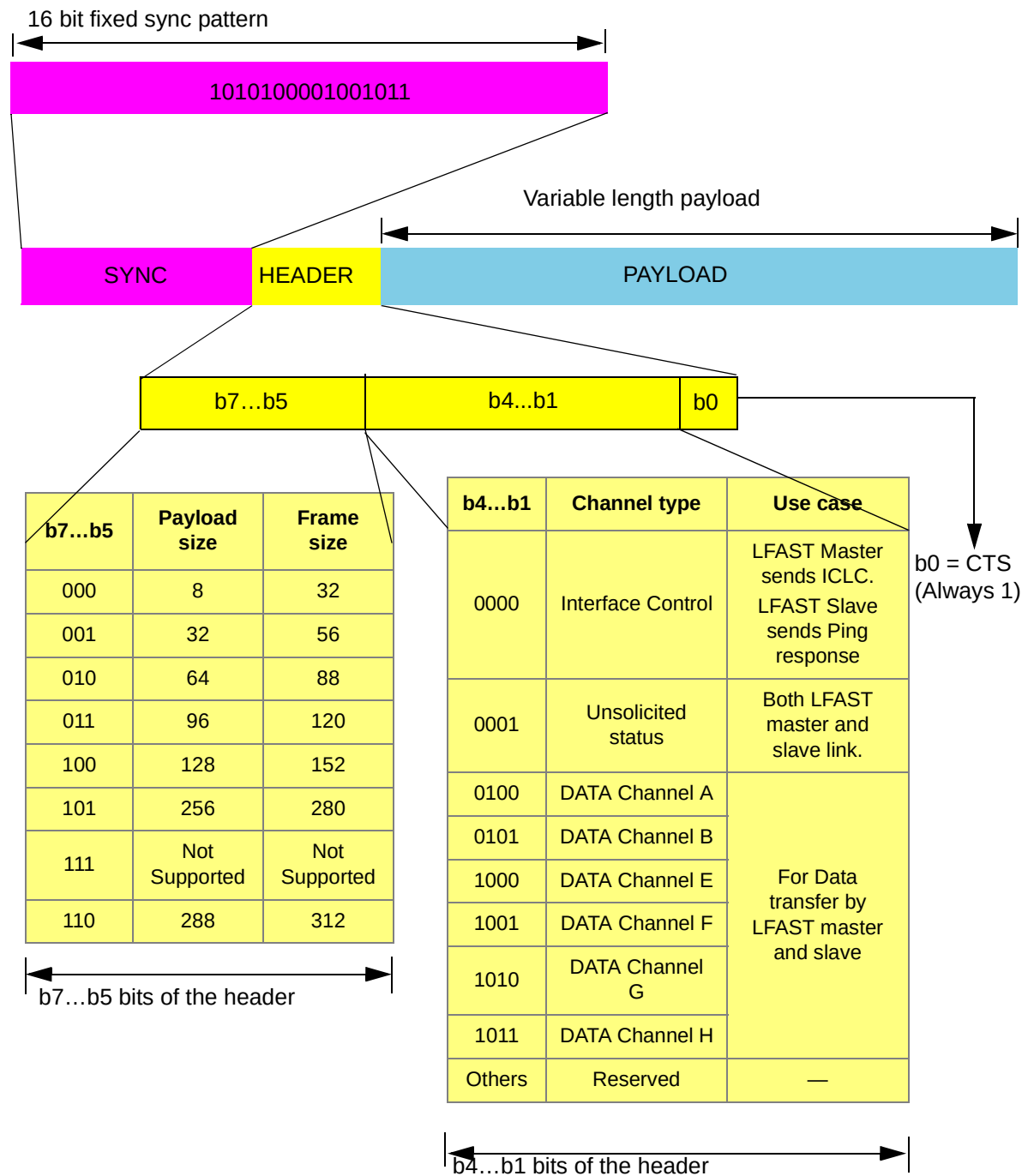


Figure 1691. LFAST frame structure

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

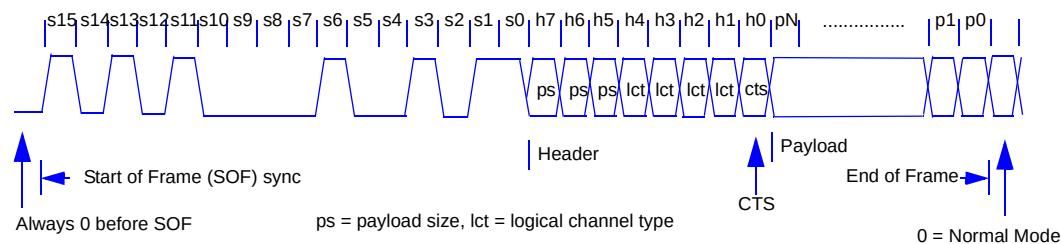


Figure 1692. Serial frame structure

The frame structure is shown in [Figure 1692](#) and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size according to [Table](#) .

Table 1576. Header payload sizes

b7-b5 (bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	288	312
111	—	—

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type.
- **Header - (b0):** CTS on the both LFAST master and slave devices. In slave only mode the CTS field will always be 1.
- **Payload: Content dependent upon frame type.**
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode).

71.5 Features

- Supports slave only mode.
- Supports asynchronous data transfer rates up to a maximum of 320 Mbps
- Transmits and receives data and unsolicited frames.
- Receives ICLC frames
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports flow control using sliding window protocol.
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation.
- Supports for detection of unsupported channel number and unsupported payload size.

71.6 Memory map and register definition

All registers are 32 bits wide.

71.6.1 Memory map

Table 1577. LFAST memory map

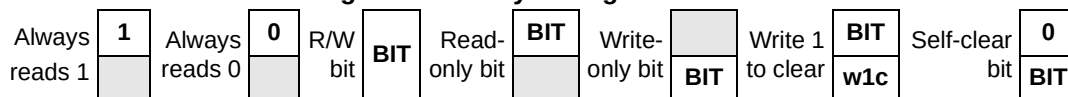
Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0000	LFAST Mode Configuration Register (MCR)	R/W	0000_0010	on page 3054
0004	LFAST Speed Control Register (SCR)	R/W	0001_0000	on page 3055
0008	LFAST Correlator Control Register (COCR)	R/W	0000_000E	on page 3056
000C	LFAST Test Mode Control Register (TMCR)	R/W	0000_0000	on page 3058
0010	LFAST Auto Loopback Control Register (ALCR)	R/W	0000_0000	on page 3059
0014	LFAST Rate Change Delay Control Register (RCDCR)	R/W	000F_0000	on page 3060
0018	LFAST Wakeup Delay Control Register (SLCR)	R/W	1201_5F02	on page 3061
001C	Reserved	—	—	—
0020	LFAST Ping Control Register (PICR)	R/W	0000_80CA	on page 3063
0024–002C	Reserved	—	—	—
0030–0038	Reserved – Access will not generate errors	—	—	—
003C	Reserved	—	—	—
0040	LFAST LVDS Control Register (LCR)	R/W	0000_5024	on page 3064

Table 1577. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0044	LFAST Unsolicited Tx Control Register (UNSTCR)	R/W	0000_0000	on page 3066
0048	LFAST Unsolicited Tx Data Registers 8 (UNSTDR8)	R/W	0000_0000	on page 3067
004C	LFAST Unsolicited Tx Data Registers 7 (UNSTDR7)	R/W	0000_0000	on page 3067
0050	LFAST Unsolicited Tx Data Registers 6 (UNSTDR6)	R/W	0000_0000	on page 3067
0054	LFAST Unsolicited Tx Data Registers 5 (UNSTDR5)	R/W	0000_0000	on page 3067
0058	LFAST Unsolicited Tx Data Registers 4 (UNSTDR4)	R/W	0000_0000	on page 3067
005C	LFAST Unsolicited Tx Data Registers 3 (UNSTDR3)	R/W	0000_0000	on page 3067
0060	LFAST Unsolicited Tx Data Registers 2 (UNSTDR2)	R/W	0000_0000	on page 3067
0064	LFAST Unsolicited Tx Data Registers 1 (UNSTDR1)	R/W	0000_0000	on page 3067
0068	LFAST Unsolicited Tx Data Registers 0 (UNSTDR0)	R/W	0000_0000	on page 3067
006C–007C	Reserved	—	—	—
0080	LFAST Global Status Register (GSR)	R	8000_0000	on page 3067
0084–0090	Reserved	—	—	—
0094	LFAST Data Frame Status Register (DFSR)	R	0000_0000	on page 3069
0098	LFAST Tx Interrupt Status Register (TISR)	R/W	0000_0000	on page 3070
009C	LFAST Rx Interrupt Status Register (RISR)	R/W	0000_0000	on page 3071
00A0	LFAST Rx ICLC Interrupt Status Register (RIISR)	R/W	0000_0000	on page 3072
00A4	LFAST PLL and LVDS Status Register (PLLLSR)	R	0002_0003	on page 3074
00A8	LFAST Unsolicited Rx Status Register (UNSRSR)	R/W	0000_0000	on page 3075
00AC	LFAST Unsolicited Rx Data Register 8 (UNSRDR8)	R	0000_0000	on page 3075
00B0	LFAST Unsolicited Rx Data Register 7 (UNSRDR7)	R	0000_0000	on page 3075
00B4	LFAST Unsolicited Rx Data Register 6 (UNSRDR6)	R	0000_0000	on page 3075
00B8	LFAST Unsolicited Rx Data Register 5 (UNSRDR5)	R	0000_0000	on page 3075
00BC	LFAST Unsolicited Rx Data Register 4 (UNSRDR4)	R	0000_0000	on page 3075
00C0	LFAST Unsolicited Rx Data Register 3 (UNSRDR3)	R	0000_0000	on page 3075
00C4	LFAST Unsolicited Rx Data Register 2 (UNSRDR2)	R	0000_0000	on page 3075
00C8	LFAST Unsolicited Rx Data Register 1 (UNSRDR1)	R	0000_0000	on page 3075
00CC	LFAST Unsolicited Rx Data Register 0 (UNSRDR0)	R	0000_0000	on page 3075

The conventions in [Figure 1693](#) and [Table 1578](#) serve as a key for the register summary and individual register diagrams.

Figure 1693. Key to Register Fields



[Table 1578](#) provides a key for register figures and tables and the register summary.

Table 1578. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bits value (other than a hardware reset).
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but its cleared by the H/W once the operation defined by the bit is over
Reset Values	
0	Resets to zero.
1	Resets to one.
[signal_name]	Reset value is determined by polarity of indicated signal.

71.6.2 Register descriptions

71.6.2.1 Mode Configuration Register (MCR)

Figure 1694. Mode Configuration Register (MCR)

Offset: 0000h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	IPGDBG	0	0	0	0	0	0	0	0
W																LSSEL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0	0	0	0	0	0	0	0	TXARBD	0	0	DRFRST ⁽¹⁾	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Set by user software and cleared by system hardware.

Table 1579. MCR field descriptions

Bits	Description
0-6	Reserved
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8-14	Reserved
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode. Refer to Section 71.10.2, Slow speed clock for more detailed description. 0 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.

Table 1579. MCR field descriptions (continued)

Bits	Description
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS LR. 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface is not disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command frame. 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 Not allowed. Do not set when in slave only mode.
19-26	Reserved
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28-29	Reserved
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after reset. 0 No soft reset. 1 Soft reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

71.6.2.2 Speed Control Register (SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Figure 1695. Speed Control Register (SCR)

Offset: 0004h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DRMD ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RDR	0	0	0	0	0	0	0	TDR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 0.

Table 1580. SCR field descriptions

Bits	Description
0-14	Reserved
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. This bit should not be modified by S/W and left as SCR[DRMD] =1.
16 - 22	Reserved
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. This bit should not be modified by S/W.
24-30	Reserved
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. This bit should not be modified by S/W.

71.6.2.3 Correlator Control Register (COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Figure 1696. Correlator Control Register (COCR)

Offset: 0008h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMPSEL ⁽¹⁾								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CORRTH ⁽¹⁾			PHSSEL ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

1. Only writable only when MCR[RXEN] = 0.

Table 1581. COCR field descriptions

Field	Description
0-7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. 00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8-27	Reserved
28-30 CORRTH	Correlator threshold level. Defines the correlation threshold level. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. 0 8 phases 1 4 phases

71.6.2.4 Test Mode Control Register (TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes

Figure 1697. Test Mode Control Register (TMCR)

Offset: 000Ch								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	CLKTST	LPON	0	0	0	0	0	LPMOD ⁽¹⁾		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFRMTH ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Table 1582. TMCR field descriptions

Bits	Description
0-5	Reserved
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by H/W on reception of an ICLC command. 1Clock Test mode enabled 0Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1Loopback mode is enabled 0Loopback mode is disabled
8-12	Reserved

Table 1582. TMCR field descriptions (continued)

Bits	Description
13-15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation Others - Reserved (should not be written by S/W)
16-31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

71.6.2.5 Auto Loopback Control Register (ALCR)

Figure 1698. Auto Loopback Control Register (ALCR)

Offset: 0010h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LPCNTEN ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Table 1583. ALCR field descriptions

Bits	Description
0-14	Reserved
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto predefined loopback frame transmission. 0 Infinite predefined loopback frame transmission enabled 1 Fixed count of predefined loopback frame transmission enabled
16-31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of predefined auto loopback frames to be sent. The predefined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload ... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

71.6.2.6 Rate Change Delay Control Register (RCDCCR)

Figure 1699. Rate Change Delay Control Register (RCDCCR)

Offset: 0014h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DRCNT ⁽¹⁾			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 1.

Table 1584. RCDCR field description

Bits	Description
0-11	Reserved
12-15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. 0h 0 cycles 1h 1 cycles ... Eh 14 cycles Fh 15 cycles
16-31	Reserved

71.6.2.7 Wakeup Delay Control Register (SLCR)

Figure 1700. Wakeup Delay Control Register (SLCR)

Offset: 0018h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HSCNT ⁽¹⁾								0	0	0	0	LSCNT ⁽¹⁾			
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HWKCNT ⁽¹⁾								0	0	0	0	LWKCNT ⁽¹⁾			
W																
Reset	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

1. Only writable when MCR[DRFEN] = 0.

Table 1585. SLCR field descriptions

Bits	Description
0-7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. 00h 0 cycle 01h 1 cycle ... 12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8-11	Reserved
0-15	Reserved
12-15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16-23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. 00h 0 cycles 01h 1 cycle ... 5Fh 95 cycles (1.18 us) ... FFh 255 cycles Maximum
24-27	Reserved
28-31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. 0h 0 cycle 1h 1 cycle 2h 2 cycles (1.18 us) ... Eh 14 cycles Fh 15 cycles

71.6.2.8 Ping Control Register (PICR)

Figure 1701. Ping Control Register (PICR)

Offset: 0020h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PNGREQ ⁽²⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PNGAUTO ⁽¹⁾	0	0	0	0	0	0	0	PNGPYLD ⁽¹⁾							
W																
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

1. Only writable when MCR[DRFEN] = 0.

2. Set by user software and cleared by system hardware.

Table 1586. PICR field descriptions

Bits	Description
0-14	Reserved
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master This bit should not be modified by S/W.
17-23	Reserved
24-31 PNGPYLD	Defines the LFAST slaves ping reply frame payload content.

71.6.2.9 LVDS Control Register (LCR)

Figure 1702. LVDS Control Register (LCR)

Offset: 0040h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SWWKLD ¹	SWSLPLD ¹	SWWKLR ¹	SWSLPLR ¹	SWOFFLD	SWONLD	SWOFFLR	SWONLR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0	0	0	0	0	LVRXOP			LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

1. Set by user software and cleared by system hardware.

Table 1587. LCR field description

Bits	Description
0-7	Reserved
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be put in sleep
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This bit can be changed only when DRFEN is high. This bit should not be modified by S/W.
13 SWONLD	SW signal to turn ON the LVDS LD. This bit can be changed only when DRFEN is high. This bit should not be modified by S/W.
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This bit can be changed only when DRFEN is high. This bit should not be modified by S/W.

Table 1587. LCR field description (continued)

Bits	Description
15 SWONLR	SW signal to turn ON the LVDS LR. This bit can be changed only when DRFEN is high. This bit should not be modified by S/W.
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable. The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21	Reserved
22-24 LVRXOP	LVRXOP[2] Termination resistance 0 Disable termination 1 Enable termination LVRXOP[1] Not used LVRXOP[0] Defines the LVDS LR Bias current reference
25-28	Reserved
29 LVTXOP	Control signal for LFAST and Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Data select. This bit is used for selecting use of clock sampled data or normal data in LVDS. 0 normal data to be used inside the LVDS 1 synchronised data to be used inside the LVDS
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

71.6.2.10 Unsolicited Tx Control Register (UNSTCR)

Figure 1703. Unsolicited Tx Control Register (UNSTCR)

Offset: 0044h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																USNDRQ ⁽²⁾
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	UNSHDR ⁽¹⁾						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0
2. Set by user software and cleared by system hardware.

Table 1588. UNSTCR field descriptions

Bits	Description
0-14	Reserved
15 USNDRQ	Tx Unsolicited send request. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16-24	Reserved
25-31 UNSHDR	Tx Unsolicited message header.

71.6.2.11 LFAST Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)

Figure 1704. Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)

Offset: 0048h (UNSTDR8)...0068h (UNSTDR0)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UNTXD[31:16] ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNTXD[15:0] ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0

Table 1589. UNSTDR_n field descriptions

Bits	Description
0-31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

71.6.2.12 Global Status Register (GSR)

Figure 1705. Global Status Register (GSR)

Offset: 0080h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0	0	0	0	0	0	0	0	0	0	0	0	LRMD	LDSM	DRSM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LPTXDN	LPFPDV	LPCLPDV	LPCHDV	LPCLSDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1590. GSR field descriptions

Bits	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module Slave only mode 1 LFAST Module in dual mode
1-12	Reserved
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active. 0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Recieve Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16-26	Reserved
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by Auto Loopback Control Register (ALCR) LPFMCNT. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

71.6.2.13 Data frame Status Register (DFSR)

Figure 1706. Data frame Status Register (DFSR)

Offset: 0094h

Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0	0	RXDCNT						0	0	0	0	0	RXFCNT			
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R			TXDCNT						0	0	0	0	0	TXFCNT			
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 1591. DFSR field descriptions

Bits	Description
0-1	Reserved
2-7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8-12	Reserved
13-15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
18-23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24-28	Reserved
29-31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

71.6.2.14 Tx Interrupt Status Register (TISR)

Figure 1707. Tx Interrupt Status Register (TISR)

Offset: 0098h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1592. TISR field descriptions

Bits	Description
0-13	Reserved
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16-26	Reserved
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28	Reserved
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

71.6.2.15 Rx Interrupt Status Register (RISR)

Figure 1708. Rx Interrupt Status Register (RISR)

Offset: 009Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1593. RISR field descriptions

Bits	Description
0-7	Reserved
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. 0 Interrupt event has not occurred 1 Interrupt event has occurred- On reception of frame with payload size for a frame other than mentioned in Table 1606 .

Table 1593. RISR field descriptions (continued)

Bits	Description
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in Table 1606 .
16-27	Reserved
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
31	Reserved

71.6.2.16 Rx ICLC Interrupt Status Register (RIISR)

Figure 1709. Rx ICLC Interrupt Status Register (RIISR)

Offset: 00A0h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ICPFF	ICPSF	ICPRE	ICTOF	ICLPE	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONE
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1594. RIISR field descriptions

Bits	Description
0-17	Reserved
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

71.6.2.17 PLL and LVDS Status Register (PLLLSR)

Figure 1710. PLL and LVDS Status Register (PLLLSR)

Offset: 00A4h								Access: User read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LRSLPS	LDLSPS	LDPDS	LRPDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 1595. PLLSR field descriptions

Bits	Description
0-27	Reserved
28 LRSLPS	This bit indicates the real time status of the LR sleep signal. 0 LR sleep signal is negated. 1 LR sleep signal is asserted.
29 LDLSPS	This bit indicates the real time status of LD sleep signal. 0 LD power sleep signal is negated. 1 LD power sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LR power down signal. When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LD power down signal. When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.

71.6.2.18 Unsolicited Rx Status Register (UNSRSR)

Figure 1711. Unsolicited Rx Status Register (UNSRSR)

Offset: 00A8h Access: read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	URXDV	0	0	0	0	0	URPCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1596. UNSRSR field descriptions

Bits	Description
0-22	Reserved
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24-28	Reserved
29-31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited data payload present in the Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0). The bitfield coding is same as b7-b5 in Table 1597 .

71.6.2.19 Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Figure 1712. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Offset: 00ACh (UNSRDR8)...00CCh (UNSRDR0) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UNRXD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1597. UNSRDRn field descriptions

Bits	Description
0-31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data. It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

71.6.3 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements^(x). All registers of the LFAST module are classified as non-safety relevant.

The MCR[IPGDBG] fulfills the safety requirements for monitoring of the debug freeze signal.

71.7 Functional description

71.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

71.7.1.1 LFAST slave interface startup procedure

1. After POR the RCDCCR are programmed according to the LVDS parameter in the device datasheet.
2. The PLLCR^(y) is programmed with the configuration parameters of the PLL.
3. The LCR is programmed with the configuration parameters of the LVDS.
4. MCR[DATAEN] fields is programmed to select the LFAST mode.
5. Write MCR[TXARBD] = 0 to enable Tx of frames.
6. Write MCR[DRFEN] = 1 to enable LFAST.
7. Write MCR[RXEN] = 1 to enable LFAST Rx.
8. On the assertion of the LFAST interface enable the signals LR power down and LR disable are negated.
9. When an ICLC frame with payload 31h is received H/W writes RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
10. The ping frame is sent when an ICLC frame with payload 00h frame is received and H/W writes RIISR[ICPRF] = 1.

x. See the "Functional Safety" chapter for register classification details.

y. The PLLCR is shared between the LFAST – Interprocessor Communications and the LFAST – High Speed Debug modules. The programming values for the PLLCR in the LFAST – Interprocessor Communications are also used by the LFAST – High Speed Debug module.

Speed mode change:

11. The PLL starts when an ICLC frame with payload 02h is received and H/W writes RIISR[ICPONF] = 1.
12. The speed of the Tx interface is changed when an ICLC frame with payload 80h is received and H/W writes RIISR[ICTFF] = 1. This will cause SRC[TDR] = 1.
13. The speed of the Rx Interface is changed when an ICLC frame with payload 10h is received and H/W writes RIISR[ICRFF] = 1. This will cause SCR[RDR] = 1.
14. The ping frame is sent when an ICLC frame with payload 00h is received and H/W writes RIISR[ICPRF] = 1.

71.7.2 LFAST Interface enable signal (lfast_sysclk_en)

The lfast_sysclk_en signal comes from the DCI module. It is asserted by the DCI internal state machine when switching from JTAG to LFAST and vice-versa can safely be done. The detailed effects of lfast_sysclk_en on the module are:

LFAST interface enable (lfast_sysclk_en) negation:

1. If LCR[SWONLD] = 0 the LD LVDS pad is powered down.
2. If LCR[SWONLR] = 0 the LR LVDS pad is powered down.
3. The LFAST slave exits the test mode (for example, loopback mode or clock test mode). Then S/W writes MCR[CLKTST] = 0 and MCR[LPON] = 0.
4. The LFAST slave Rx and Tx interfaces are set to Low Speed mode. Then write SCR[TDR] = 0 and SCR[RDR] = 0.
5. The Tx and Rx interface controllers are disabled.
6. The status registers are cleared.

LFAST interface enable (lfast_sysclk_en) assertion:

1. The LD LVDS pad is power up, if all conditions are met:
 - LCR[SwoffLD] = 0.
 - LCR[SWONLD] = 1.
 - MCR[TXEN] = 1.
 - When frame ICR[ICLCPLD] = 31h is received.
2. LR LVDS pad is powered up if all conditions are met:
 - LCR[SwoffLR] = 0.
 - LCR[SWONLR] = 1.
 - MCR[RXEN] = 1.

71.7.3 Line Receiver (LR)**71.7.3.1 Introduction**

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 1713 shows the block diagram of Uplink Controller.

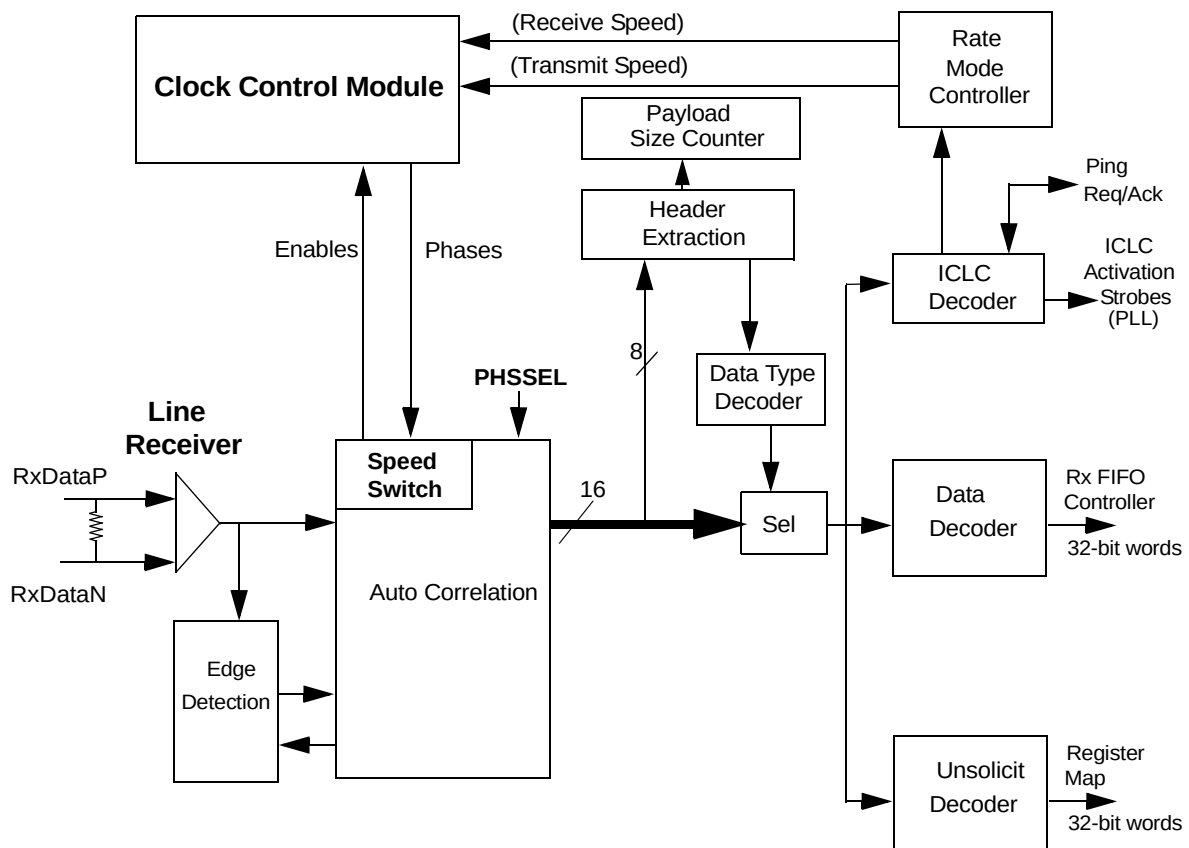


Figure 1713. Top Level Receive Controller

71.7.3.2 Edge-Detection & Auto correlation

The edge-detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge-detection circuitry is used or not. The edge-detection will be performed first if required before auto-correlation.

71.7.3.2.1 Auto correlation modes

71.7.3.2.1.1 Hunt correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the

synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

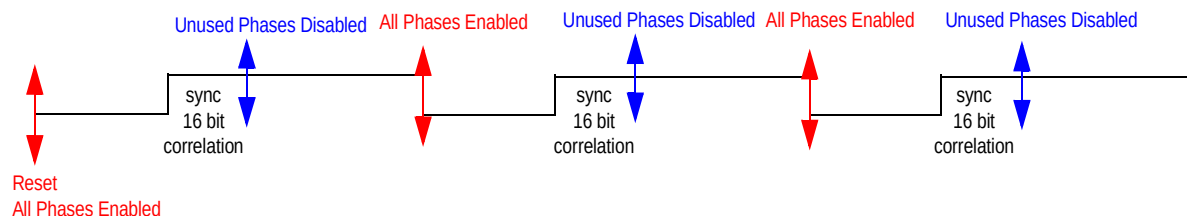


Figure 1714. Hunt Correlation mode

71.7.3.2.2 Edge detection

The edge detector is disabled in Hunt Correlation mode.

71.7.3.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in [Figure 1715](#).

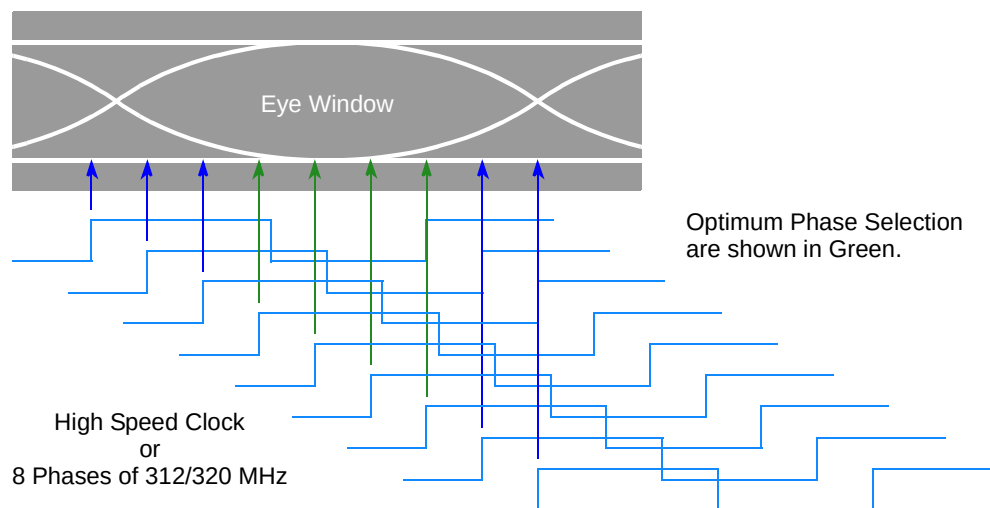


Figure 1715. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in [Figure 1716](#).

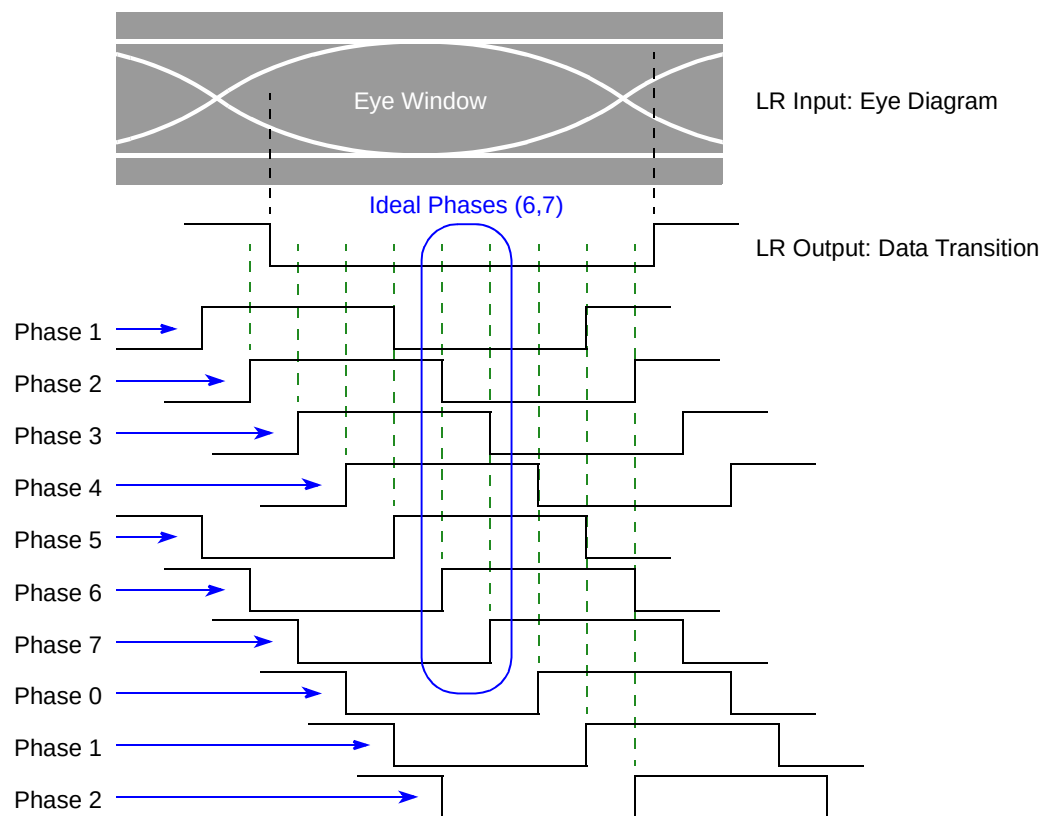


Figure 1716. High Speed 8 Phase Clock Alignment Example

Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in [Figure 1717](#) and [Figure 1718](#).

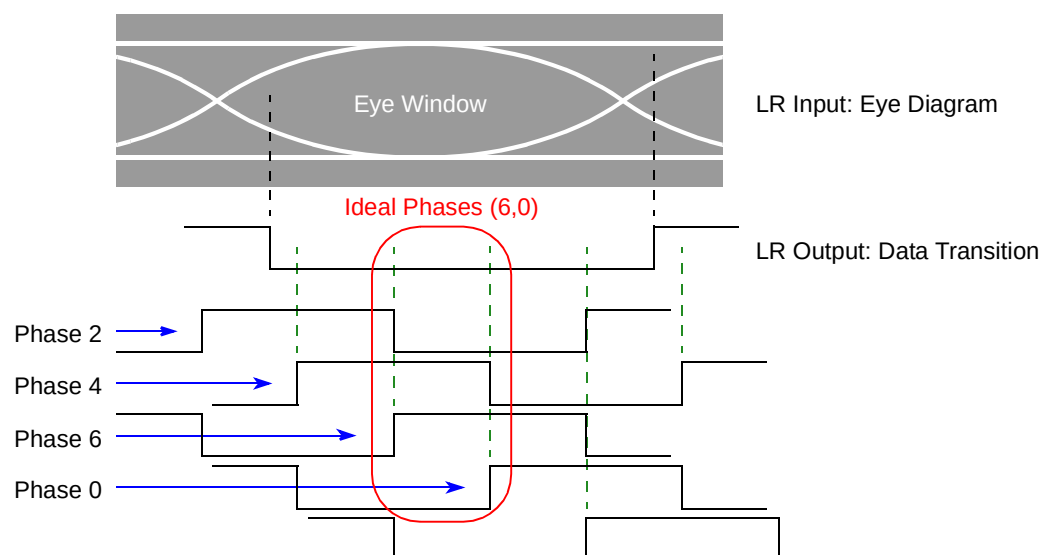


Figure 1717. High Speed 4 Phase Clock Alignment Example

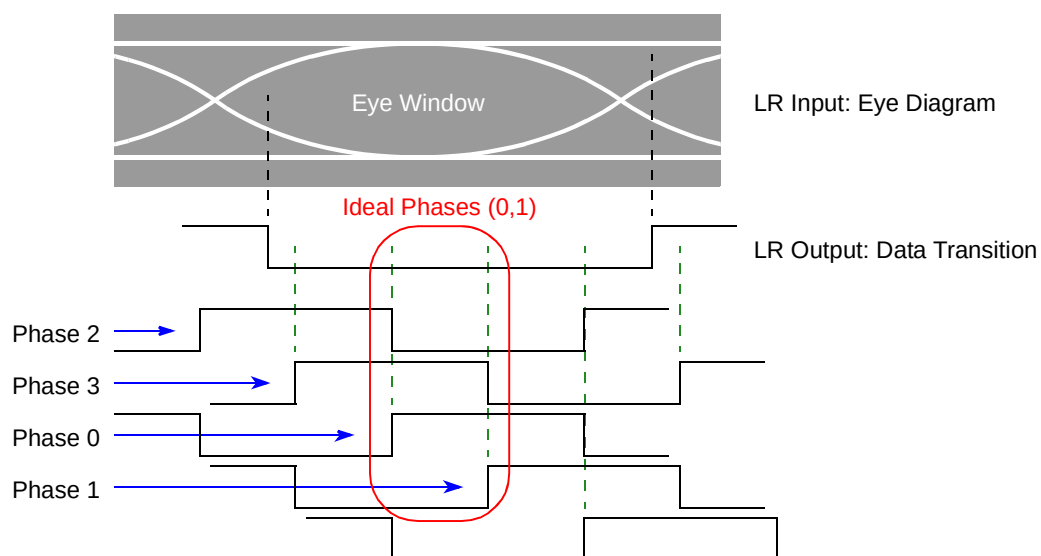


Figure 1718. Low speed 4 phase clock alignment example

The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed (312/320 MHz) 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High Speed 8 Phases: Samplers 0,1,2,3,4,5,6,7
- High Speed 4 Phases: Samplers 0,2,4,6
- Low Speed 4 Phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and re-samples it to a intermediate phase as shown in [Table 1598](#), [Table 1599](#), and [Table 1600](#) before re-sampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 1598. High speed 8 phase selection - sampling procedure

Samplers	Initial Sample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0

Table 1598. High speed 8 phase selection - sampling procedure (continued)

Samplers	Initial Sample	Intermediate Sample	Final Sample
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 1599. High speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See [Table 1599](#), Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 1600. Low speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See [Table 1600](#), Samplers 4, 5, 6, 7 are disabled. The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

71.7.3.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in [Figure 1719](#).

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (e.g Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by de-asserting the respective phase enables.

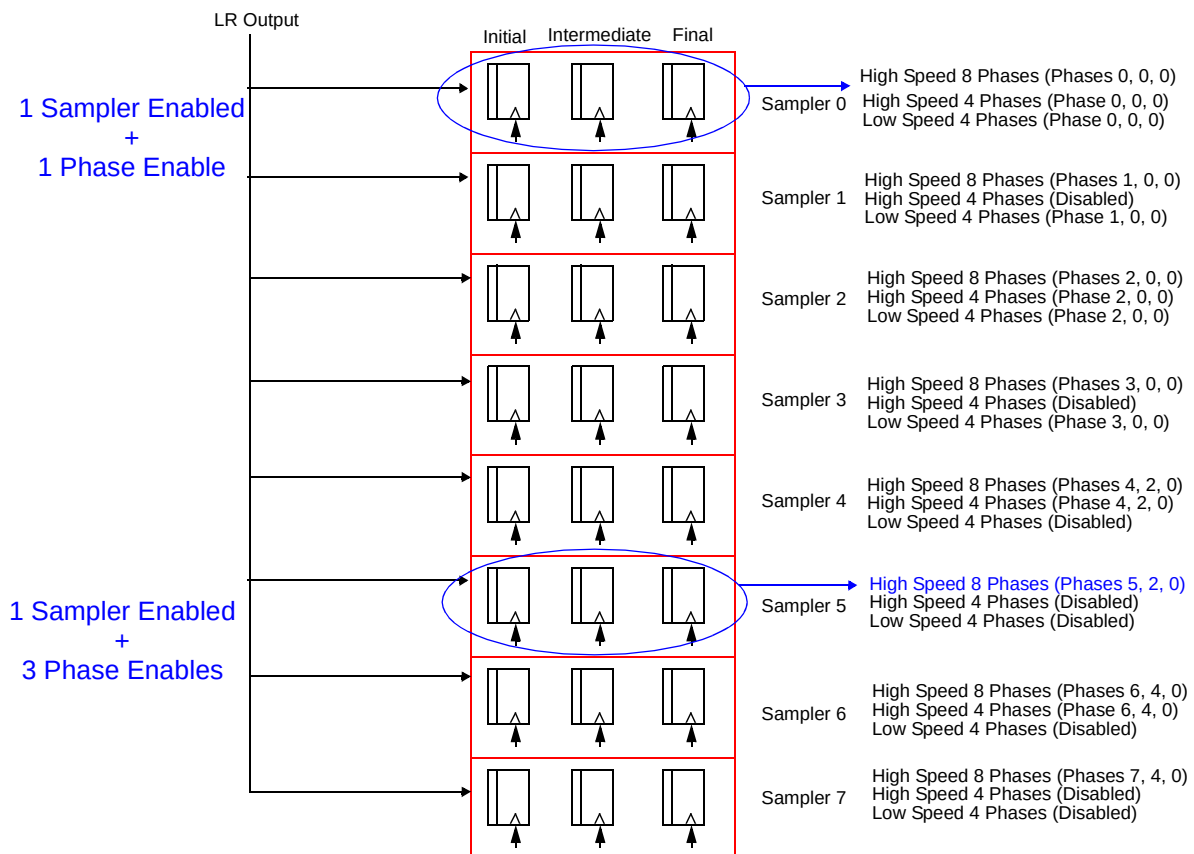


Figure 1719. Samplers, each Sampler has 3 Registers

In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in [Figure 1720](#).

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

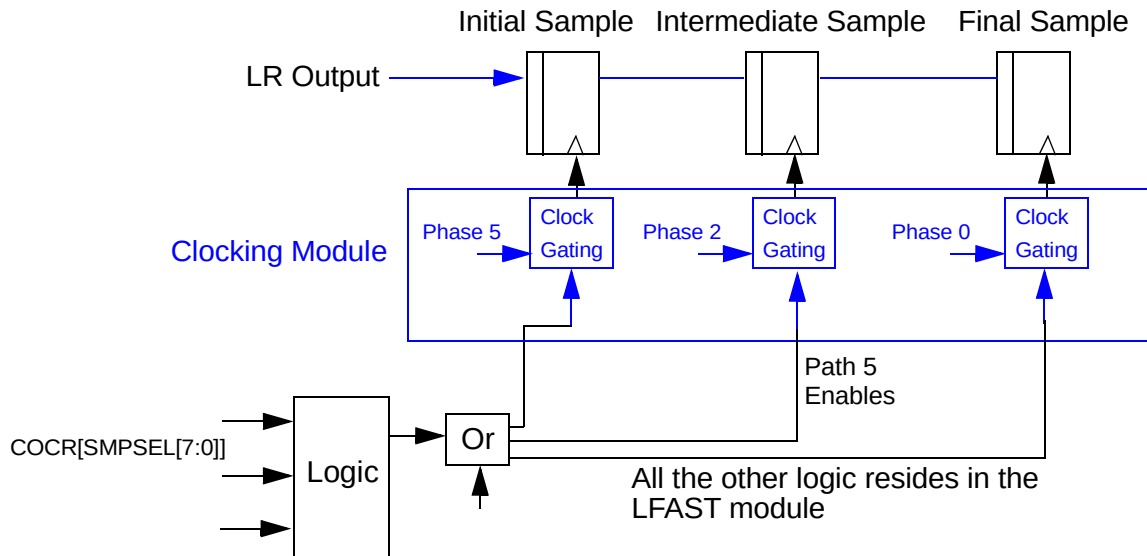


Figure 1720. Sampler 5 Logic

71.7.3.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in [Figure 1721](#)

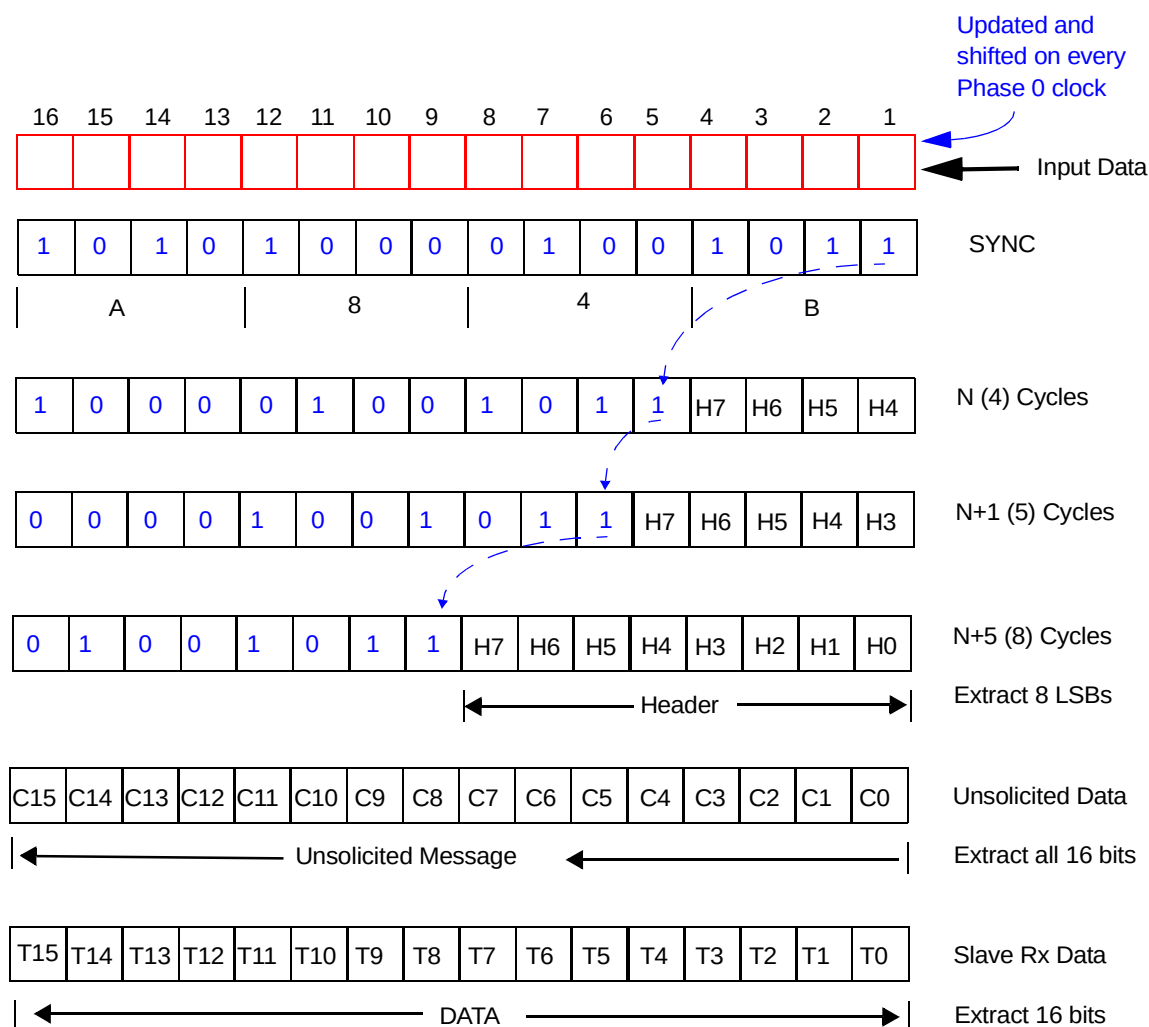


Figure 1721. Extracting Header and Payload from the 16-bit output from Auto correlation.

71.7.3.3.1 Line Receiver states

The LD has the following states:

- Shutdown**
 The LR enters in the shutdown state when the power down signal is high. In this state the LR regulator is supplying a V_{dd} level, however the LR is not enabled. In this state the LR holds its off value. Once power down signal is negated, a predefined settling time is required before the LR may be used for communication. During the settling time the peer LFAST device should not start its transfer.
- Sleep**
 The LR enters in the Sleep state when the sleep signal is high. In the sleep state, the LR is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In this

mode the receiver current consumption is reduced. Hysteresis in the LR ensures that it continues to present a 1 to the LFAST module.

- **Normal**
In the Normal state, the LR is primed for reception. Receiver converts differential signals to a single ended logic signal.

Table 1601. Line Receiver State

LR State	LFAST Slave Triggers (at least one of the following)
Shutdown	– LFAST interface enable assertion/negation – Programing of LCR[SwoffLR] or LCR{SWONLR} – Programing MCR[RXEN] and MCR[TXEN]
Sleep	– Programing of LCR[SWSLPLR] or LSC[SWWKLR]
Normal	Absence of above mentioned conditions

71.7.4 Transmit Controller

71.7.4.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, ping etc. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Recieve Controller and the ping response from the register block.

The Tx interface has two speed modes:

- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in [Figure 1722](#).

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

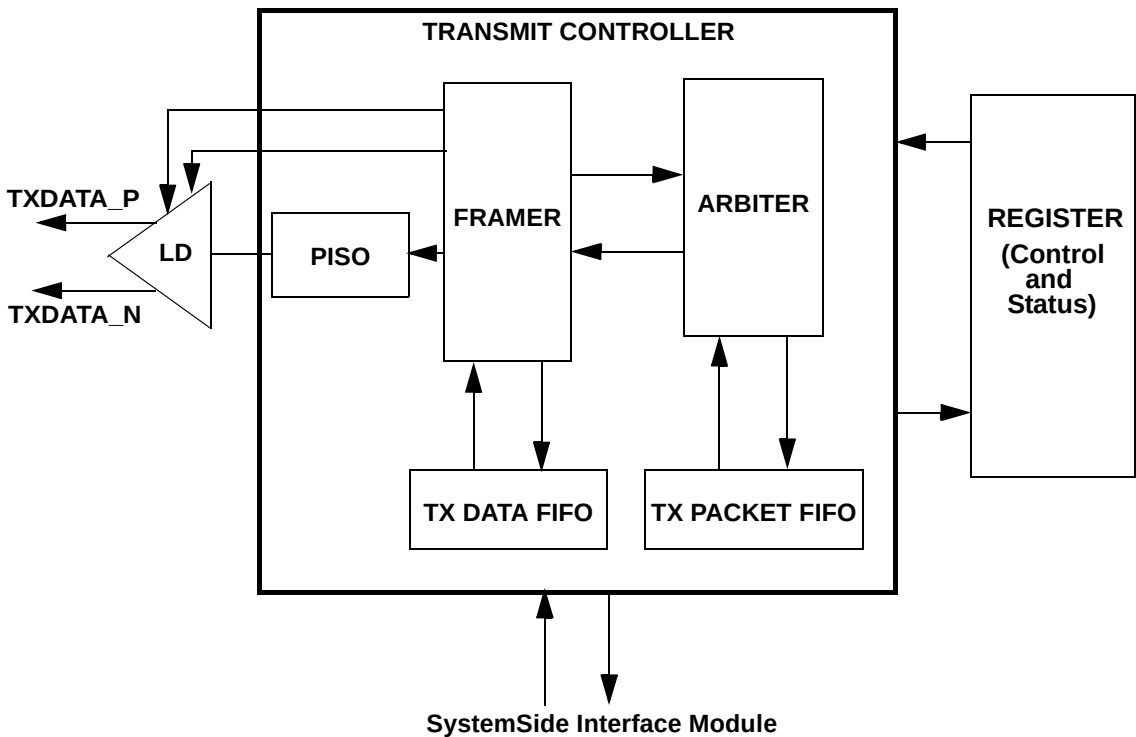


Figure 1722. Transmit Controller Connections

71.7.4.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like:

- S/W programmable registers
- System side interface FIFO
- Rx interface controller
- LFAST Slave: LFAST interface enable (lfast_sysclk_en)

The level of priority for each request is shown in [Table 1602](#).

Table 1602. Priority Levels for the Tranmsit Controller

Request	LFAST Slave Priority	Triggering Conditions (at least one of the listed)
LFAST interface enable (lfast_sysclk_en)	1	– LFAST interface enable (lfast_sysclk_en) is asserted – LFAST interface enable (lfast_sysclk_en) is negated
Tx Interface disabled	2	– MCR[DRFEN] = 0 – MCR[TXEN] = 0 – MCR[DRFRST] = 1 – MCR[TXARBD] = 1 – LFAST Slave: ICLC frame with payload for “Disable Rx interface” received

Table 1602. Priority Levels for the Transmit Controller

Request	LFAST Slave Priority	Triggering Conditions (at least one of the listed)
Tx Interface speed mode change	3	– SCR[TDR] is modified – LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	– TMCR[LPON] = 1 – LFAST Slave: ICLC frame with payload for loopback mode enable received <i>Note:</i> Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
Ping response request	5	– LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 – LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	– Valid only if Last Frame with header b0 = 1 received from the peer LFAST – UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	– Tx FIFO contains one or more frames – Last Frame with header b0 = 1 received from the peer LFAST device – MCR[DATAEN] = 1
Clock mode test	8	– TMCR[CLKTST] = 0 – LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCCR[DRCNT].

71.7.4.3 Line Driver digital connections

71.7.4.3.1 Line Driver states

The LD has the following states:

- **Shutdown**
The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling

time is required before the LD may be used for communication. The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In the sleep state the LD outputs are connected to V_{cm} (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a predefined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programmed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

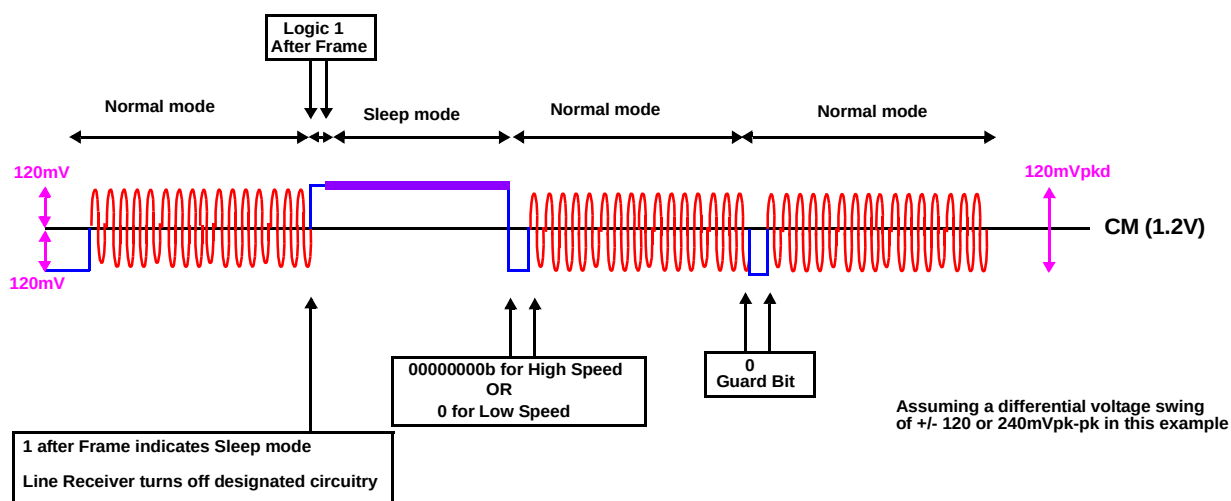


Figure 1723. Example of Sleep mode

As shown in [Figure 1723](#) before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 1603. Line Driver States

LD State	LFAST Slave Triggers
Shutdown	– LFAST interface enable (lfast_sysclk_en) negation/assertion – Programing LVDS[SWOFFLD] and LVDS[SWONLD] – ICLC command from LFAST master – Programing MCR[TXEN] and MCR[DRFEN]
Sleep	– Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions

71.7.5 Frames supported

[Table 1604](#) provides the frames supported and their permissible payload sizes

Table 1604. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx 2. LFAST Slave Rx	Supported by 1. LFAST Slave Tx 2. LFAST Master Rx	Payload Size (bits)
Data Frame	Rx – 0100b, 1000b – 1011b Tx – All	YES	YES	TX: 96 RX: 128
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO ⁽¹⁾	8
CTS Frame	n/a	NO	NO	n/a
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

71.7.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

71.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

71.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO. Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

71.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTE] = 1 after each frame transfer.

71.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[TXEN] and MCR[DRFEN] bits should be set to enable the data frame reception. When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded.

71.7.6.2 Unsolicited flow

71.7.6.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

71.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
 - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
 - UNSTCR[USNDRQ] = 0
 - TISR[TXUNSF] = 1.
 - If UNSTCR[USNDRQ] = 0 then:
 - Program the unsolicited payload in UNSTD0–UNSTD8, and can be written up to a payload of frame of a maximum of 288 bits.
 - Program the unsolicited frame header in UNSTCR[UNSHDR].
 - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.

71.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDRO–UNSTD8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.

71.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is defined as a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

71.7.6.3.1 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

71.7.6.3.1.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF] = 1. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

71.7.6.3.1.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

71.7.6.3.1.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

71.7.6.3.1.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

71.7.6.3.1.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

71.7.6.3.1.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

71.7.6.3.1.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable (lfast_sysclk_en).

71.7.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

71.7.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programmed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 1605. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

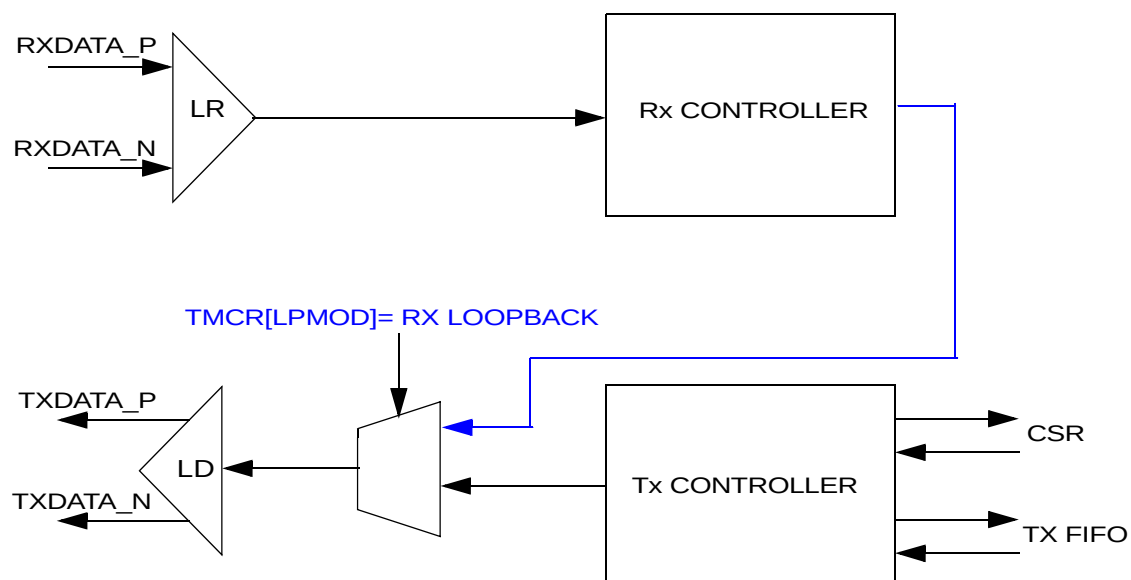
In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

71.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. Bit b0 of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

Figure 1724. Rx LoopBack mode



Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master
- For LFAST Slave: - Deassertion of the LFAST interface enable (lfast_sysclk_en).

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

71.7.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via “Rx LVDS LoopBack Mux”. Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

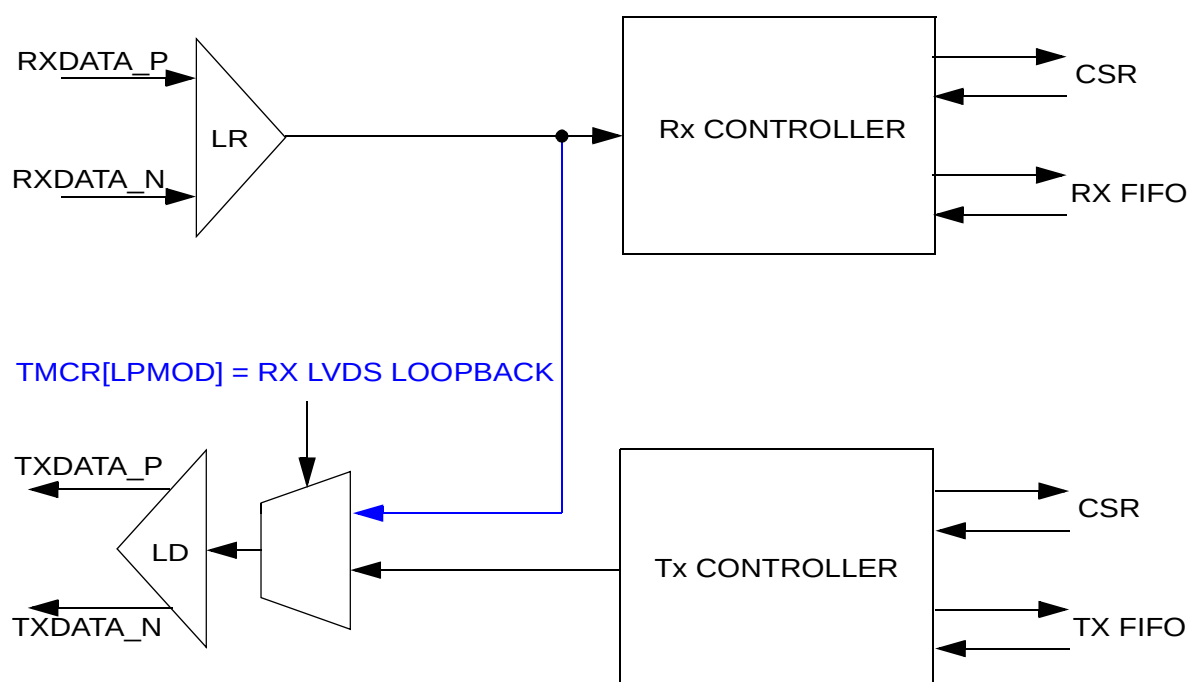


Figure 1725. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:
 - S/W writes TMCR[LPON] = 1.
 - For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
 - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.
 - Deassertion of the LFAST interface enable (lfast_sysclk_en).

71.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

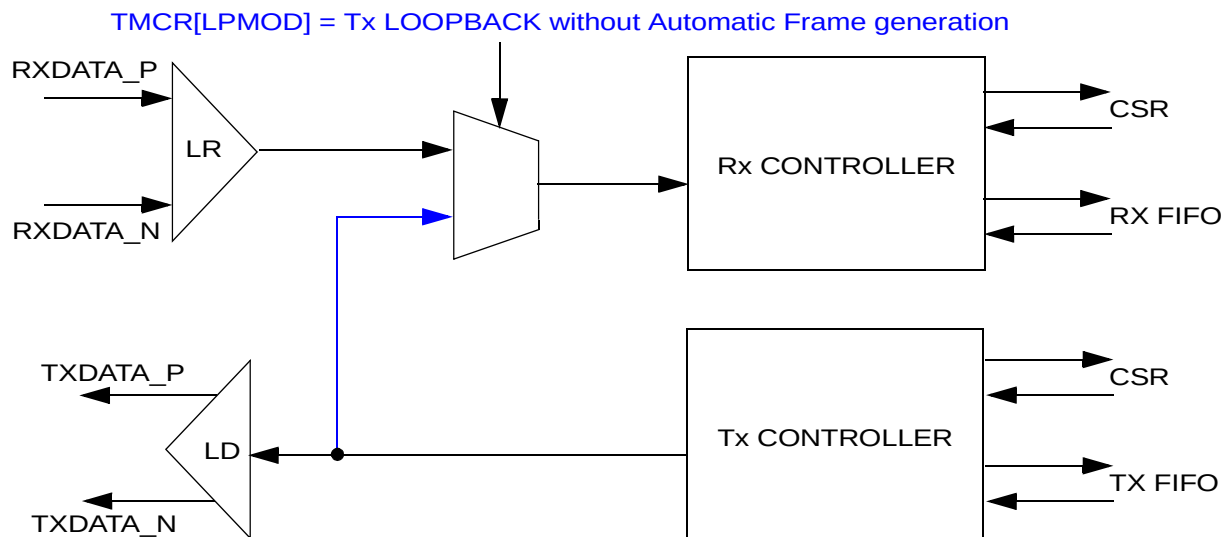


Figure 1726. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs $\text{TMCR}[\text{LPON}] = 0$.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

All frames and error flags are decoded in this mode.

71.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where predefined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- $\text{ALCR}[\text{LPCNTEN}]$ defines whether fixed number of auto loopback frames to be transmitted.
- $\text{ALCR}[\text{LPFMCNT}]$ defines the number of loopback frames to be transmitted if $\text{ALCR}[\text{LPCNTEN}] = 1$.

The Status signals and register used for these modes are:

- $\text{GSR}[\text{LPCSDV}]$: Valid Synchronization received.
- $\text{GSR}[\text{LPCHDV}]$: Frame with Header of 13h received.
- $\text{GSR}[\text{LPCPDV}]$: Frame with Payload of CBh received.
- $\text{GSR}[\text{LPTXDN}]$: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to $\text{ALCR}[\text{LPFMCNT}]$.

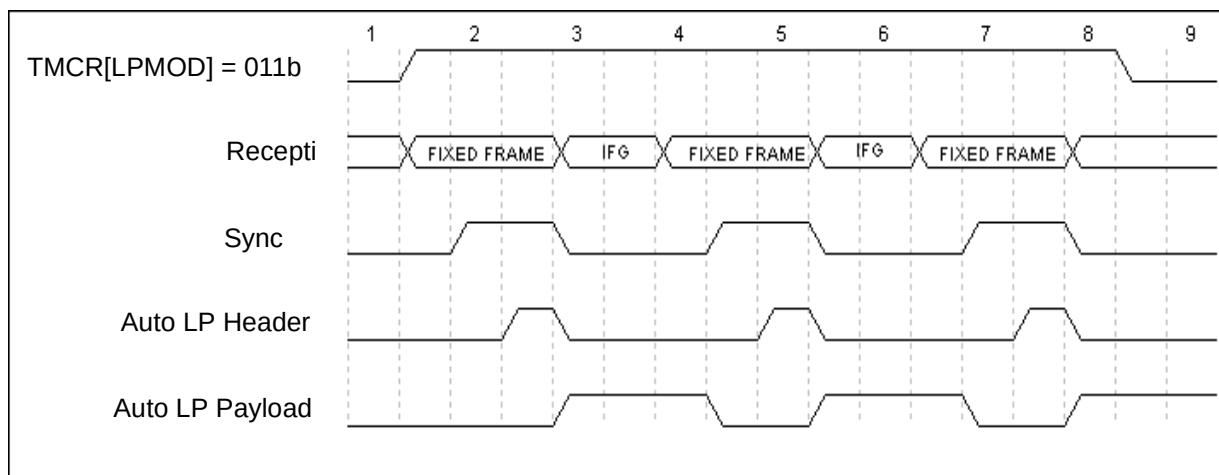


Figure 1727. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

- Tx Loopback with Automatic frame generation.
- Tx LVDS (external) with Automatic frame generation.

71.7.7.1.4.1 Tx LoopBack mode with automatic frame generation

When $TMCR[LPMOD] = 011b$ (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

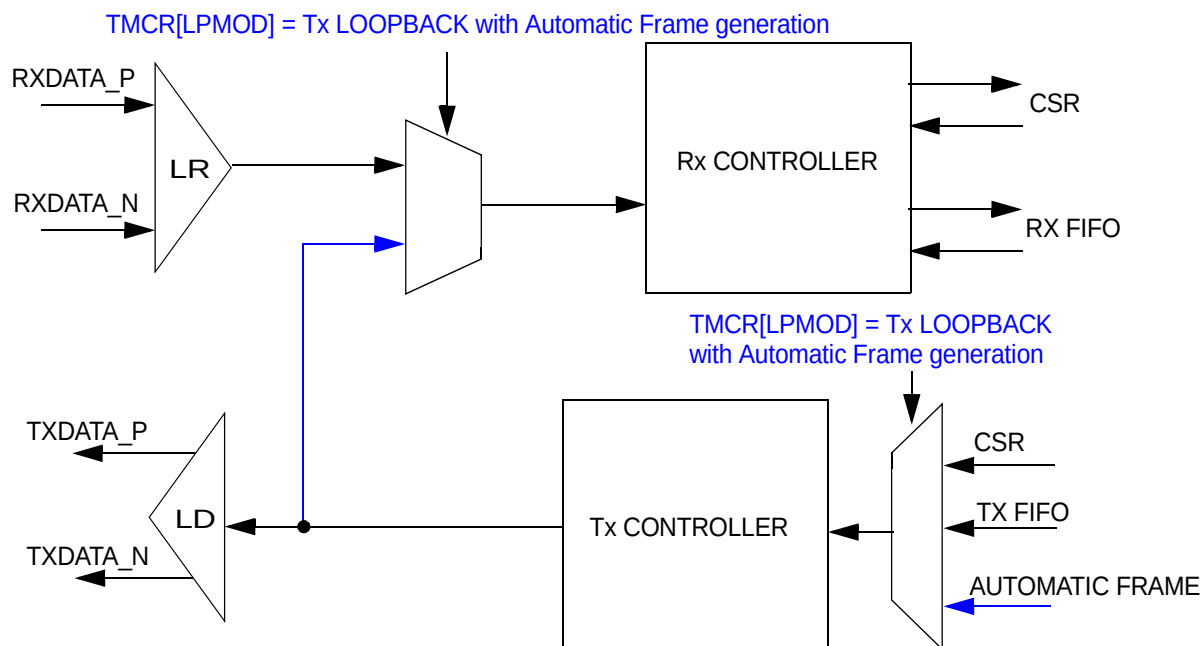


Figure 1728. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs $TMCR[LPMOD] = 011b$.
2. Loopback can be turned on by either of the following methods:
 - S/W programs $TMCR[LPON] = 1$.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from the LFAST Master.

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs $TMCR[LPON] = 0$.
- For LFAST Slave: Deassertion of the LFAST interface enable ($lfast_sysclk_en$).

71.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

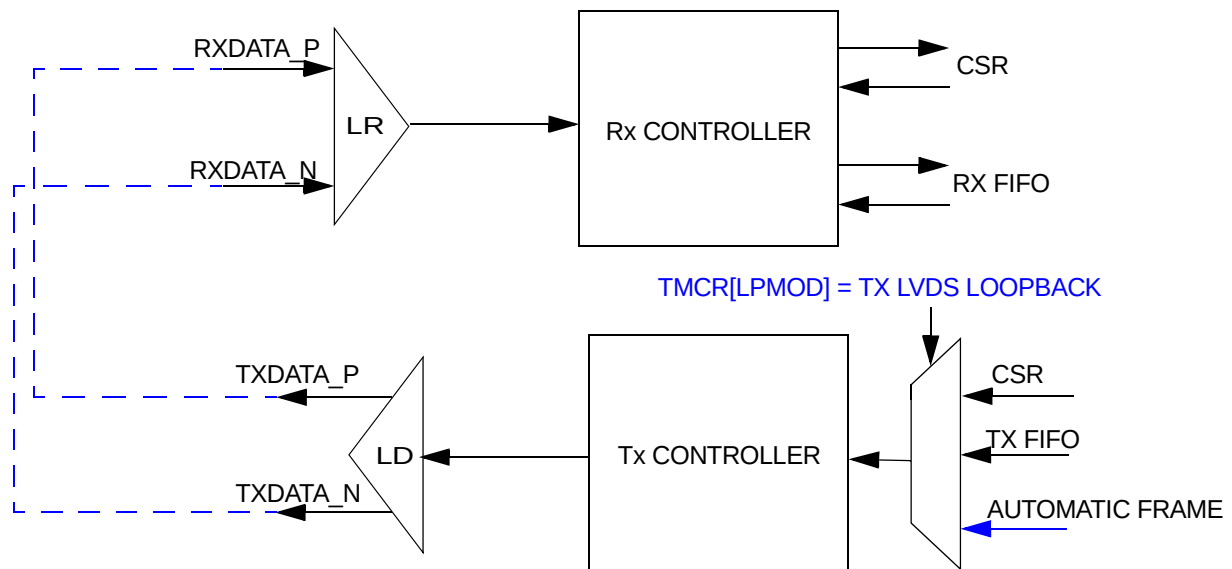


Figure 1729. Tx LVDS Loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on) from LFAST Master

Exit from Tx LVDS Loopback with automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

71.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 34h (Clock Test mode on) from LFAST Master.

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by any of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode does not affect the functionality of the Rx Interface.

71.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has its own dedicated buffer, and they are not shared between each other.

Table 1606. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer (in bits)	Rx Buffer (in bits)	Memory Type
Data Frame	6 × 32 max 2 packets	8 × 32 max 2 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	N/A	1 × 8 max 1 packet ⁽¹⁾	Registers PICR[PNGPYLD]
CTS Frame	N/A	N/A	N/A

1. Ping response packet

71.9 Resets

The various blocks in LFAST are reset as described in [Table 1607](#).

Table 1607. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ]
LFAST interface enable (lfast_sysclk_en)	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers except the following: TISR, RISR, RIISR, GSR[LPTXDN] and GSR[LPFPDV] These bits remain reset while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: Negedge only – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ] These bits does not remain reset while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: positive edge only – TISR, – RISR, – RIISR, – GSR[LPTXDN] and GSR[LPFPDV] These bits does not remain reset while the LFAST interface enable (lfast_sysclk_en) is High.

71.10 Clocks

The LFAST mainly works on two clocks:

- System Bus Clock, used to program the registers.
- System Side Module clock, which is synchronous to the System Bus clock.

71.10.1 Clocking strategy

[Figure 1730](#) shows the clock domain in which each module functions. The Clock Module will generate four phases of slow speed clock using both the edges of `lfast_sysclk`, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

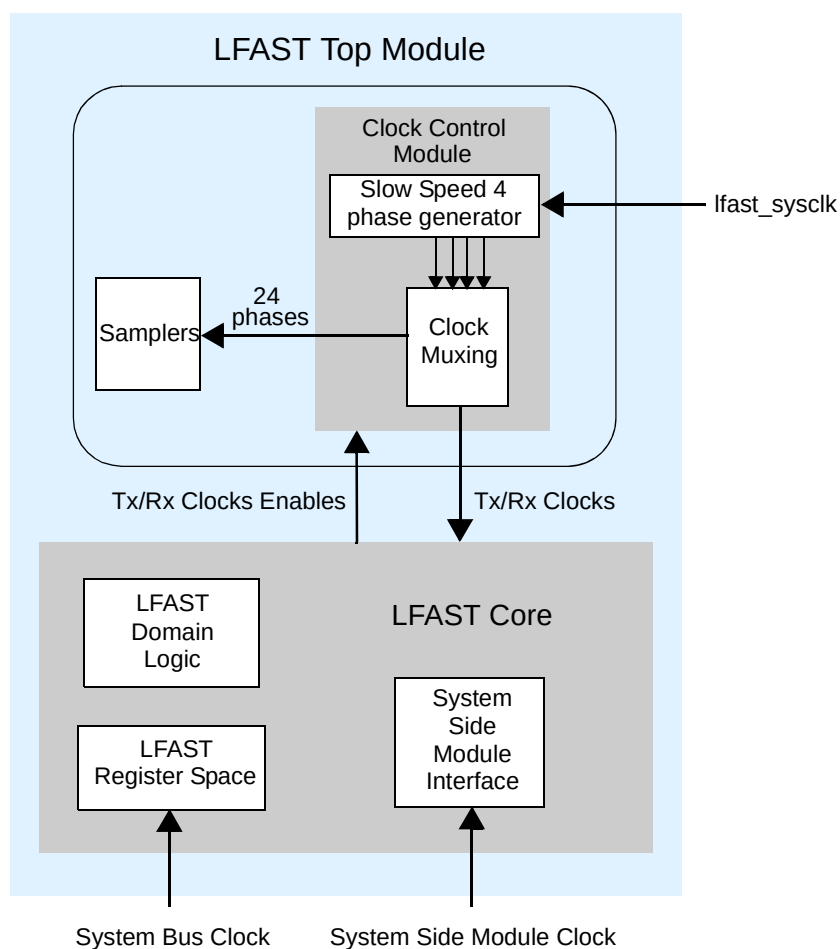


Figure 1730. LFAST module clock domains

71.10.2 Slow speed clock

71.10.2.1 External muxing

The Clock Control Module will receive `lfast_sysclk` from the clocking subsystem as shown in [Figure 1731](#), [Figure 1732](#) and [Figure 1733](#).

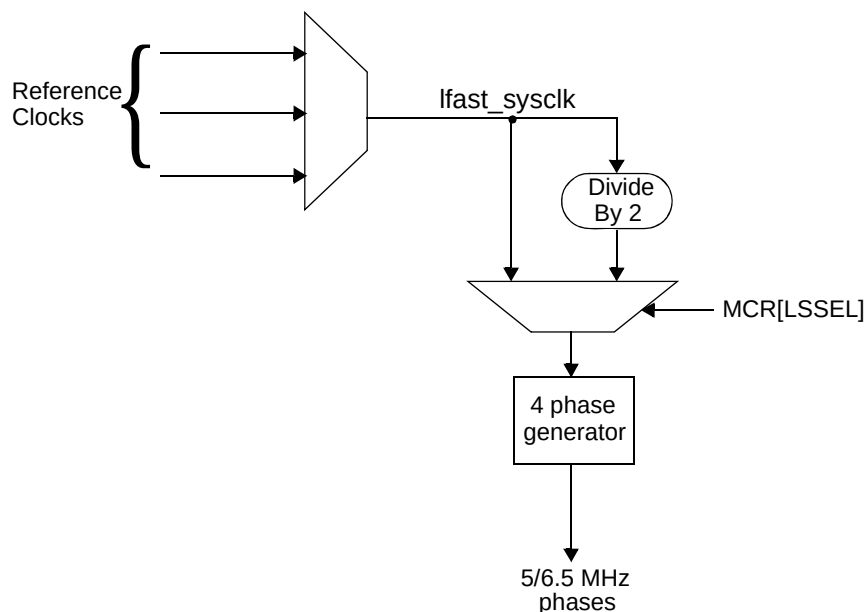
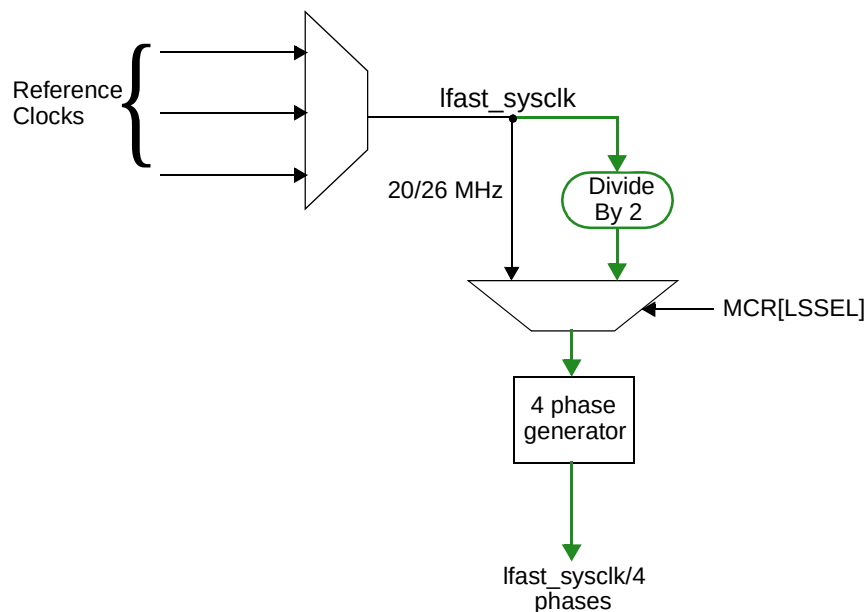


Figure 1731. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in [Figure 1731](#).

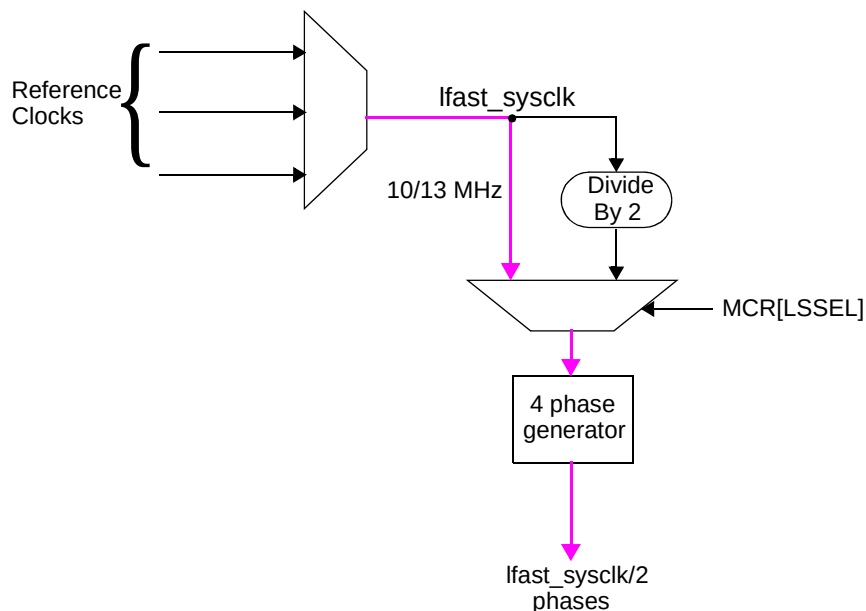
The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in [Figure 1732](#) (selected clock path shown in green).

Figure 1732. External clock muxing of lfast_sysclk of 20/26 MHz



In this case, `lfast_sysclk` is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency $\text{lfast_sysclk}/2$ when `MCR[LSSEL] = 0`. If `lfast_sysclk` is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in [Figure 1733](#) (selected clock path selected shown in magenta).

Figure 1733. External clock muxing of lfast_sysclk of 10 MHz



71.10.2.2 Slow speed 4-phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated.

[Figure 1734](#) Shows 4 phases getting generated using lfast_sysclk.

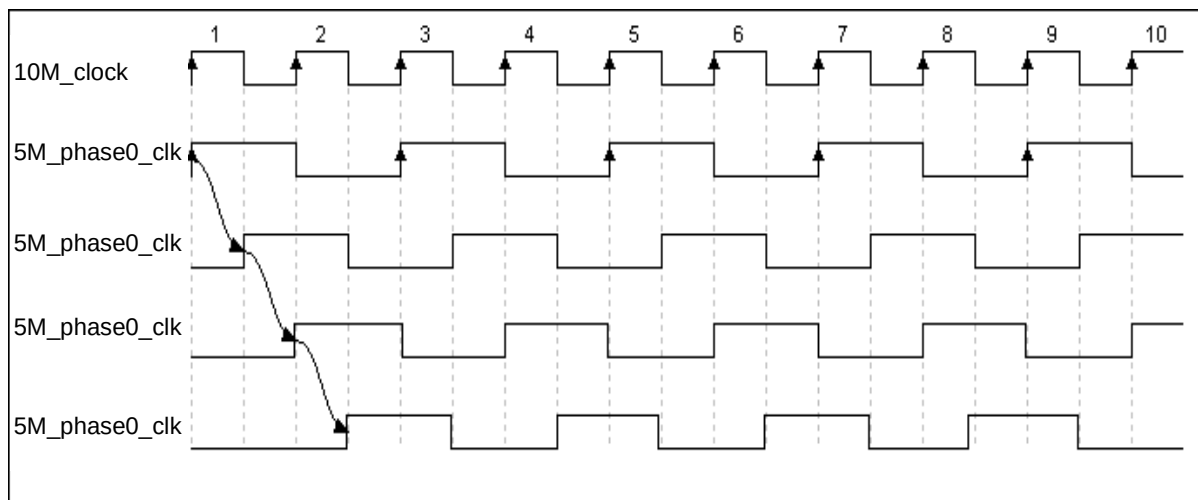


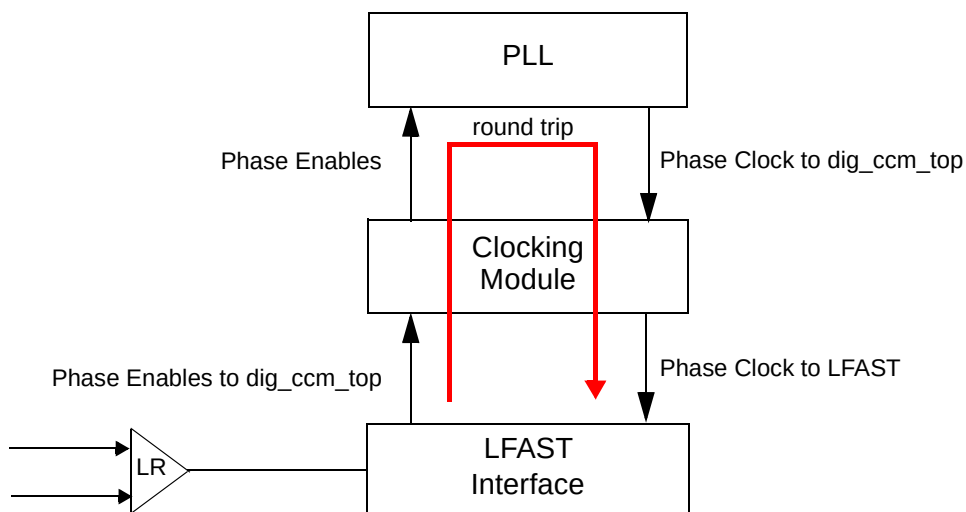
Figure 1734. Slow speed 4 phases generation

71.10.3 Rx controller clocks

71.10.3.1 Clocking module requirements for high speed phases

The high speed Phases are obtained from the PLL via the Clocking Module as shown in [Figure 1735](#).

Figure 1735. Clock Enables and Clock Paths



The LFAST block will know which of the phases will be required for the different modes of operations. Therefore, the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

71.10.3.2 Clock module requirements for low speed phases

71.10.3.2.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq $\times$ 8 MHz clock from the PLL is not needed.

Table 1608. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

71.10.4 Tx controller clocks

71.10.4.1 Clocking module requirements for speed phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See [Table 1609](#).

Table 1609. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

71.10.4.2 Transmit clock muxing

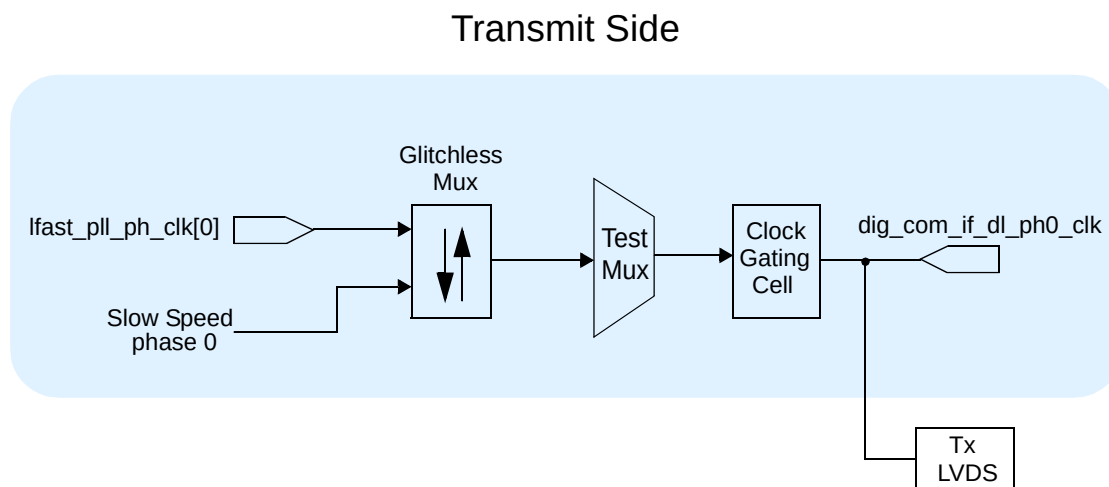


Figure 1736. Transmit Clocks Muxing

71.10.4.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

72 Development Trigger Semaphore (DTS)

72.1 Introduction

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note: When used as a component of a triggered data acquisition system, Nexus read/write access is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus Auxiliary port.

72.2 Overview

The Development Trigger Semaphore (DTS) module consists of registers and a small amount of combinational logic to generate an output signal—DTS Trigger Output (DTS_OUT). The registers are as follows:

- DTS_SEMAPHORE register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding DTS_EN bit is set in the DTS_ENABLE register. This enables an external tool to detect up to 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.
- DTS_SEMAPHORE_B register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding DTS_EN_B bit is set in the DTS_ENABLE register. This enables an external tool to detect an additional 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.
- DTS_STARTUP register—This register provides a mechanism for the external tool to notify software running on the CPU that the tool is connected and can provide information about either the type of tool or options that can be used by the software.
- DTS_ENABLE register—This register provides an enable/disable capability for the DTS feature.

The architecture is shown in [Figure 1737](#).

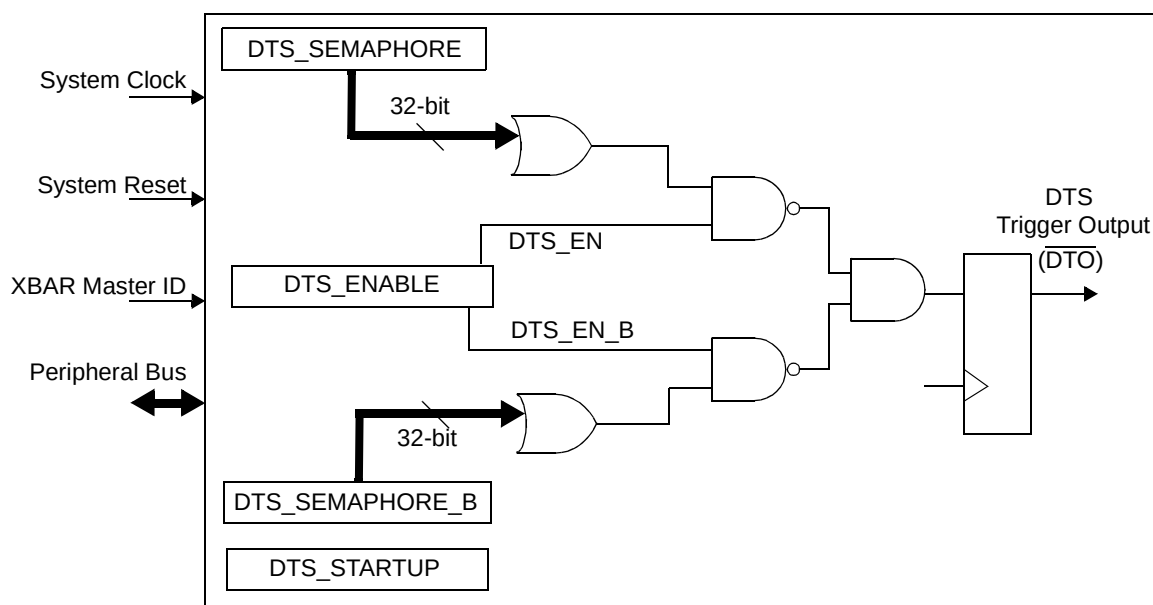


Figure 1737. DTS block diagram

The DTS Trigger Output ($\overline{DTSO}$) signal is connected to one of the $\overline{EVTO}$ inputs of the Debug and Calibration Interface (DCI). The other $\overline{EVTO}$ inputs are connected to the other Nexus modules in the device. $\overline{DTSO}$ is asserted when any bit in a semaphore register is set and the DTS function of that register is enabled.

Note: When the DTS module is enabled ($DTS_ENABLE[DTS_EN] = 0b1$ or $DTS_ENABLE[DTS_EN_B] = 0b1$), all other Nexus $\overline{EVTO}$ functions associated should be disabled by the tool and $\overline{EVTO}$ becomes the $\overline{DTSO}$. Unlike the $\overline{EVTO}$ function that only asserts for one clock, the $\overline{DTSO}$ function remains asserted until the tool reads the semaphore register that is the source of the assertion, clearing the register's contents.

Figure 1738 shows an example of the chain of events that begins with setting of a bit in the DTS_SEMAPHORE register and the clearing of the register caused by a Nexus read.

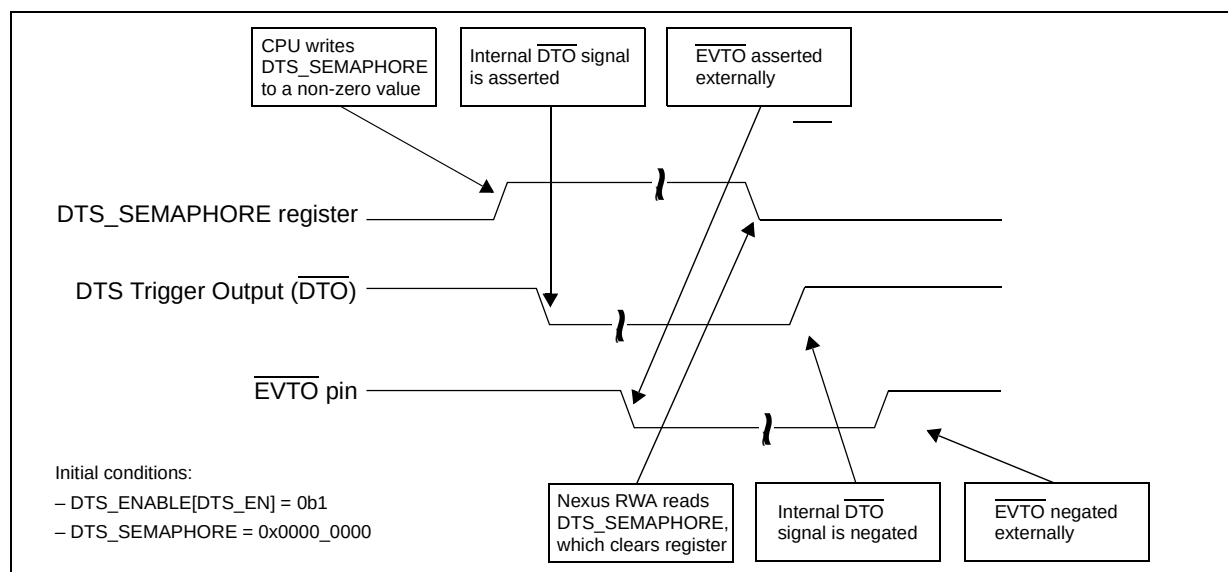
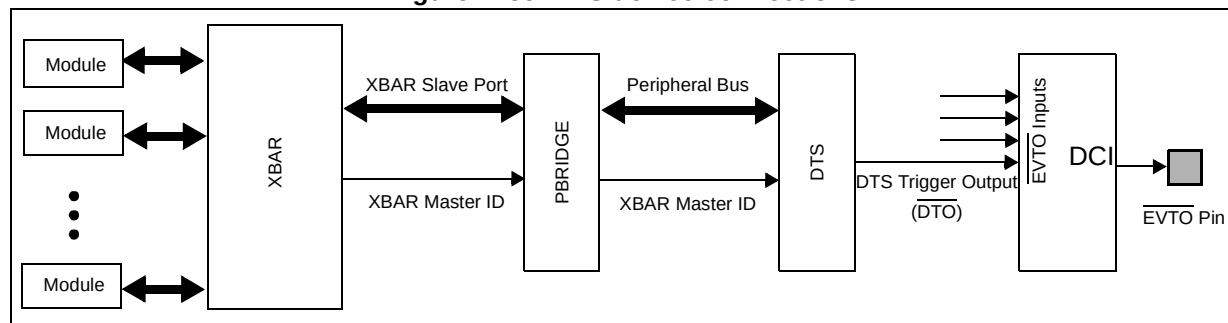


Figure 1738. DTSO event sequence example

72.3 DTS device connections

Figure 1739 shows the DTS device connections. The DTS module connects to the Peripheral Bridge (PBRIDGE) for access to the registers. The PBRIDGE is connected to the device modules through a slave port of the Crossbar bus interface (XBAR).

Figure 1739. DTS device connections



The registers have limited access as described in [Section 72.3.1, DTS register access](#). Access is based on the XBAR Master ID of the accessing module. Writing to the semaphore registers is limited to the cores and the eDMA module and is restricted to only setting bits. Only an access via a Nexus Read/Write Access from an external tool can clear bits in the semaphore registers (semaphore register bits are cleared automatically when read via Nexus/Write Access). Similarly, the DTS_ENABLE and DTS_STARTUP registers can only be written via a Nexus Read/Write Access.

Note: Nexus Read/Write Accesses use the core load/store bus to perform accesses, but Nexus accesses have a different Master ID than normal core load/stores.

72.3.1 DTS register access

A summary of accesses to all DTS registers by bus masters is provided in [Table 1610](#). Only proper 32-bit accesses are valid. The effects of write accesses that are not 32 bits are not defined.

Table 1610. DTS register access effects

Register	32-bit read				32-bit write			
	RWA ⁽¹⁾	Core	eDMA	FlexRay ⁽²⁾	RWA ⁽¹⁾	Core	eDMA	FlexRay ²
DTS_ENABLE	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_STARTUP	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_SEMAPHORE	Data and Clear ⁽³⁾	Data	Data	Data	No effect	Bit OR	Bit OR	No effect
DTS_SEMAPHORE_B	Data and Clear ⁽³⁾	Data	Data	Data	No effect	Bit OR	Bit OR	No effect

1. Nexus Read/Write access via an external tool

2. FlexRay or other master

3. A read of the semaphore registers by Nexus Read/Write Access module is destructive and clears all bits in the register

Access to DTS module registers is controlled based on the XBAR Master ID of the accessing module.

Note: *The XBAR Master ID should not be confused with the Master Port number of the XBAR. See [Chapter 13, Crossbar Switch \(XBAR\)](#), for details.*

Tools must access the DTS registers through the Nexus Read/Write Access mechanism of a core. External tool accesses through either core appear as if the access is via the core and therefore do not have the same level of access as a Nexus Read/Write Access.

72.4 Memory map and register definition

[Table 1611](#) shows the memory map of the DTS module registers.

Table 1611. DTS memory map

Offset	Register	Access	Reset Value	Section/Page
0x0	ENABLE—DTS Output Enable Register	Restricted R/W ⁽¹⁾	0x0000_0000	72.4.1.1/3112
0x4	STARTUP—DTS Startup Register	Restricted R/W ⁽¹⁾	0x0000_0000	72.4.1.2/3112
0x8	SEMAPHORE—DTS Semaphore Register	Restricted R/W ⁽¹⁾	0xFFFF_FFFF	72.4.1.3/3113
0xC	SEMAPHORE_B—DTS Semaphore Extension	Restricted R/W ⁽¹⁾	0xFFFF_FFFF	72.4.1.4/3114

1. Only certain types of accesses are allowed. See the register figure note of each register for a description of the accesses that are allowed.

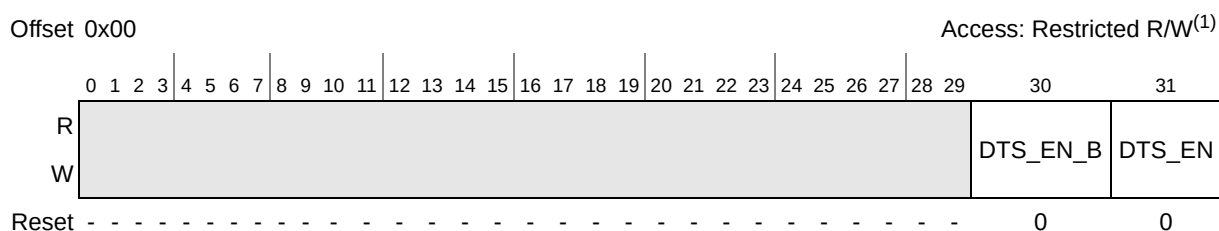
72.4.1 Register descriptions

72.4.1.1 DTS output enable register (ENABLE)

The DTS_ENABLE register controls the DTS Trigger Output ($\overline{DTO}$) and whether $\overline{DTO}$ is active on the $\overline{EVTO}$ output pin of the device. [Figure 1740](#) shows the format of the DTS_ENABLE register.

Note: Access to the DTS_SEMAPHORE, DTS_SEMAPHORE_B, and DTS_STARTUP registers are unaffected by the state of this register.

Figure 1740. DTS_ENABLE register



1. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA).

The DTS_ENABLE register is described in [Table 1612](#).

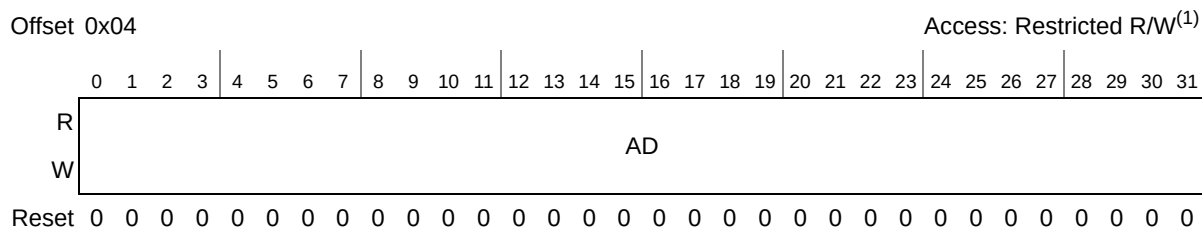
Table 1612. DTS_ENABLE register field descriptions

Field	Description
DTS_EN	DTS Enable. Controls whether the $\overline{DTO}$ signal is routed to the $\overline{EVTO}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{RESET}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE register asserts the DTS trigger output signal (DTO).
DTS_EN_B	DTS Enable B. Controls whether the $\overline{DTO}$ signal is routed to the $\overline{EVTO}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{RESET}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE_B register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE_B register asserts the DTS trigger output signal (DTO).

72.4.1.2 DTS startup register (STARTUP)

[Figure 1741](#) shows the format of the DTS_STARTUP register, used for tool detection and startup information exchange between external data acquisition tool and the embedded controller.

Figure 1741. DTS_STARTUP register



1. The DTS_STARTUP register can be read by the core, the eDMA module and Nexus but can only be updated by a Nexus Read Write Access (RWA).

The DTS_STARTUP register is described in [Table 1613](#).

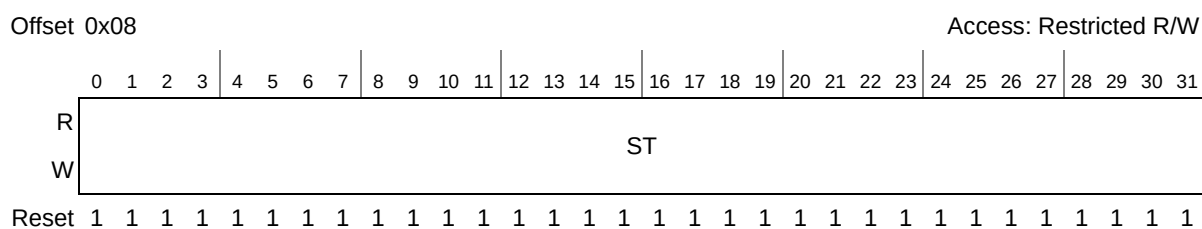
Table 1613. DTS_STARTUP register field descriptions

Field	Description
AD	Application Dependent register bits. The bits have no defined meaning to the microcontroller. They are used to by an external tool to pass information (for example, application options and status) to application software running on target microcontroller at startup time. Use a Nexus RWA 32-bit write access to update the contents of this register. A device reset (either from the $\overline{\text{RESET}}$ pin or an internally generated reset) clears all bits in the register. A Nexus reset does not change the contents of the register.

72.4.1.3 DTS semaphore register (SEMAPHORE)

[Figure 1742](#) shows the format of the DTS_SEMAPHORE register. The DTS_SEMAPHORE register is used by software to assert the DTO signal on the device EVTO pin. A one in any bit of this register causes the DTO signal on the EVTO pin to be driven low. The intended use of this register is for the DTO signal to notify tools that data is available. Individual bits are used to identify the specific data.

Figure 1742. DTS_SEMAPHORE register



The DTS_SEMAPHORE register is described in [Table 1614](#).

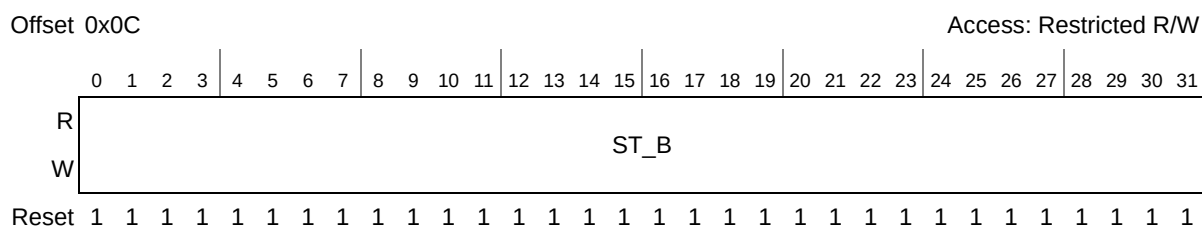
Table 1614. DTS_SEMAPHORE register field descriptions

Field	Description
ST	Semaphore Trigger. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. – All register bits are set to '1' by a device reset. – A Nexus reset does not change the state of this register. – The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. – For the core or DMA, only 32-bit write or read accesses are valid. – A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

72.4.1.4 DTS semaphore extension (SEMAPHORE_B)

[Figure 1743](#) shows the format of the DTS_SEMAPHORE_B register. The DTS_SEMAPHORE_B register is used by software to assert the DTO signal on the device EVTO pin. A one in any bit of this register causes the DTO signal on the EVTO pin to be driven low. The intended use of this register is for the DTO signal to notify tools that data is available. Individual bits are used to identify the specific data.

Figure 1743. DTS_SEMAPHORE_B register



The DTS_SEMAPHORE_B register is described in [Table 1615](#).

Table 1615. DTS_SEMAPHORE_B register field descriptions

Field	Description
ST_B	Semaphore Trigger Extension. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. – All register bits are set to '1' by a device reset. – A Nexus reset does not change the state of this register. – The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. – For the core or DMA, only 32-bit write or read accesses are valid. – A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

72.5 Example application

The calibration process of a new engine requires real-time access to calibration tables and the ability to update the tables in real-time. The DTS module enables this capability by enabling software to assert a signal to an external device pin to notify an external tool that data is available. The tool can then retrieve the data.

In this type of application the DTS_SEMAPHORE register and DTS Trigger Output (DTO) signal provide a mechanism to notify the calibration tool that the calibration variable or variables (or sets of measurements), up to 32, have been updated with new values and are available for the tool to access.

Note: It is the user's responsibility to ensure that the tool has time to retrieve the data prior to that particular trigger being set a second time. It is also permissible to have multiple triggers active at the same time or for a second trigger to be set before a previous trigger has been serviced, as long as it is not the same trigger (unless it is acceptable to the tool to not receive every data set).

Figure 1744 shows an example DTS startup sequence for an external real-time data acquisition system. The startup and synchronization sequence can be as simple or as complicated as the need requires. However, a typical startup sequence is as follows:

1. The DTS_STARTUP register is cleared by a power on reset or any CPU reset.
2. The tool writes a non-zero value to the DTS_STARTUP register.
3. The CPU (user application software) then reads the value of the DTS_STARTUP register. Based on this value, different initialization options can be selected. The bits can be used for any application specific definitions.
4. Since the DTS_SEMAPHORE register is cleared when the tool reads the current value. The tool should perform all necessary initialization before reading this register. The application software can then check that the DTS_SEMAPHORE register was cleared by the tool, to determine that it is safe to start using it for its intended raster trigger semaphore function.
5. An optional hand shake from the CPU can be used to inform the tool that the user software has detected that the tool is attached and the CPU has performed the proper initialization for the tool by writing a predefined value to the DTS_SEMAPHORE register (the example shown in the figure above uses 0xAAAA_AAAA—all A's was used since it is unrealistic that 16 channels could be enabled very quickly after start up after a reset).

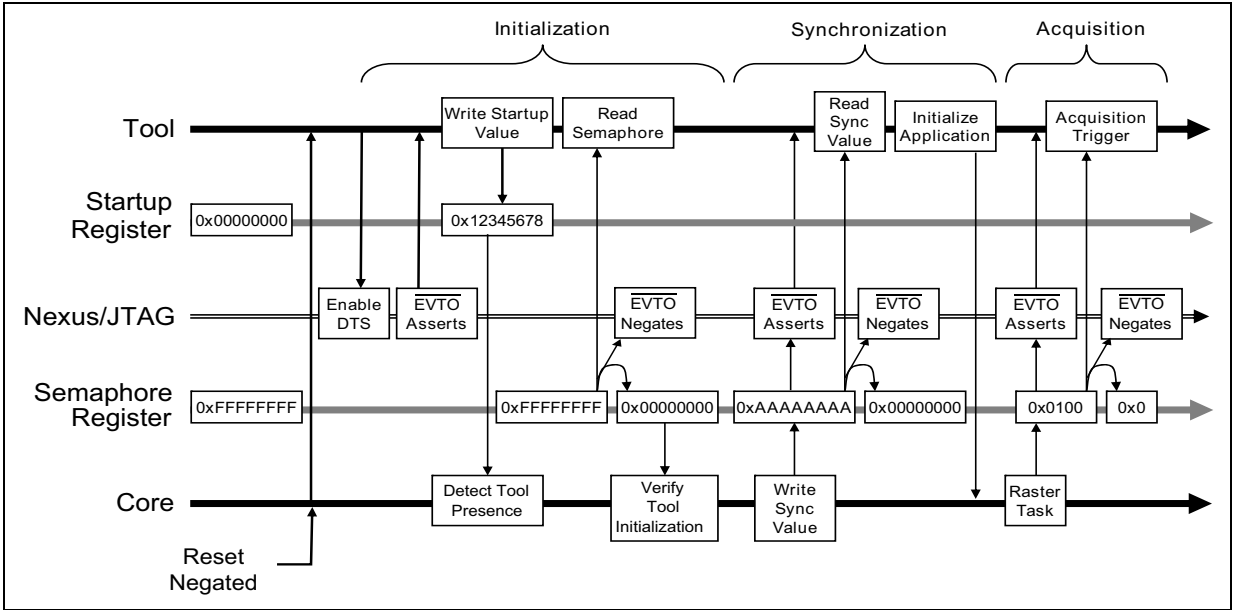


Figure 1744. DTS startup sequence example

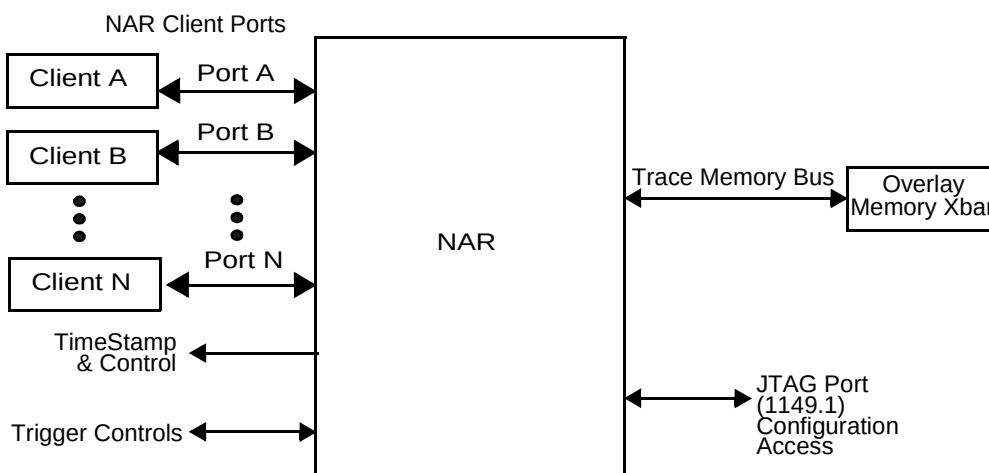
73 Nexus Aurora Router (NAR)

73.1 Introduction

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

Top-level architecture is shown in [Figure 1745](#). Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Figure 1745. NAR top level block diagram



The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. The Production Device (PD) does not support partitioning of the main queue; therefore, only one of the ports can be active at a time. Data going through the Aurora out port or the Trace port goes through an additional translation block to put the data in the correct format. The actual Aurora formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The NAR supports the following client interfaces:

- Legacy interface to support existing client interface protocol through a conversion gasket logic with clock domain transfer
- New client interface with clock domain transfer

73.2 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port and/or the Trace port. The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (1149.1) port.

73.3 Features

The NAR supports the following features:

- Connects 7 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Legacy client interface for backward compatibility
- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Time-stamping provides a central synchronization resource for message ordering. Gray coded timer output for clients working in different clock domains.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client
- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message
- Event generation on External Trace Memory (overlay) programmable full
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

73.4 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 1616. NAR registers

Register name	Read/ Write	JTAG index (NAR relative)	Location
NAR_CR—NAR Control Register	R/W	0	on page 3119
NAR_ST—NAR Status Register	R	1	on page 3121
NAR_SFR—Source Filter Register	R/W	2	on page 3123
NAR_TFR—Type Filter Register	R/W	3	on page 3124
NAR_TBALO—Trace memory port base address, low	R/W	4	on page 3125
NAR_TBAHI—Trace memory port base address, high	R/W	5	on page 3125
NAR_TCR—Trace Memory bus control register	R/W	6	on page 3127
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)	on page 3128
NAR_CSSR—Client Suppress Status Register	R	B	on page 3129
NAR_CDR—Client Disable Register	R/W	C	on page 3130
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E	on page 3131

73.4.1 Register descriptions

73.4.1.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following figure shows the format of the NAR_CR.

Figure 1746. NAR control register (NAR_CR)

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG		SED	SRD	TBW	AQP		0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	0	0	0	ALT		0	AFA	0	0	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CR is described in the following table.

Table 1617. NAR_CR field descriptions

Field	Description
31 NEN	NAR Enable. 0 NAR is disabled unless HBDP is independently enabled. See Section 73.5.2.2, Disabled for more information. 1 NAR is enabled.
30 NCD	NAR Communication Disable. 0 NAR monitors clients for activity 1 NAR communication with clients is disabled
29 TEN	Timestamp Enable. 0 Timestamp packet is not included in messages 1 Timestamp packet is included in messages from all client including NAR generated messages
28 TRS	Timestamp Reset. 0 to 1 transition: reset timestamp counter in NAR
27–26 TSG	Timestamp Granularity (Mode). <i>Note: Some clients may only support timestamps on every message.</i> 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 32nd message
25 SED	Stall Error Disable. 0 NAR injects dummy complete message when active client stalls 1 NAR does not inject dummy complete message
24 SRD	Stall Recovery Disable. 0 Stalled receive queue resumes operation when its client comes back on line 1 Stall error recovery is disabled
23 TBW	Tracebuffer Wrap. 0 NAR does not capture new data when main queue is full 1 NAR begins overwriting main queue from the beginning upon queue overflow
22–21 AQP	Alternate Queue Partition. 00 Queue partitioning is disabled 01 Reserved 10 Reserved 11 Reserved
20	Reserved and must be written with 0
19–16	Reserved and must be written with 0s
15–13 WPE	Watchpoint Enable 000 NAR watchpoint messaging is disabled xx1 NAR generates a watchpoint message upon a sync event x1x NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event 1xx NAR generates a watchpoint message upon a suppress-mode event
12	Reserved
11	Reserved. This bit must be written with 0.
10–9	Reserved and must be written with 0s.

Table 1617. NAR_CR field descriptions (continued)

Field	Description
8	Reserved
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications. 00 No alternate port. Running in trace-buffer mode. 01 Aurora out is the ALT 10 Trace memory bus is the ALT 11 Reserved
5	Reserved
4 AFA	Automatic Flush Accumulators. 0 Output queue accumulators do not flush stranded beats 1 Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.
2	Reserved and must be written with 0.
1 PUS	Power-Up Suppressed. 0 NAR powers up active 1 NAR powers up in global suppressed mode
0 NSR	NAR Soft Reset. 1 Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

73.4.1.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following figure shows the format of the NAR_ST.

Figure 1747. NAR status register (NAR_ST)

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

The NAR_ST register is described in the following table.

Table 1618. NAR_ST field descriptions

Field	Description
31 NEN	NAR Enabled. 0 NAR is disabled and inactive 1 NAR is enabled with clocks on
30–28 NUM	NAR Mode. 000 NAR is in disabled mode 001 NAR is in global suppressed mode 010 NAR is in communication-disabled mode 011 NAR is in tracebuffer mode 100 NAR is in active transmit mode 101 Reserved 110 Reserved 111 NAR is in run-control mode
27–24 NCE	NAR Config Error. See Table 1632 for coding
23–16	Reserved
15 HPR	HBDP Ready. 0 HBDP (Aurora Out port) is not ready 1 HBDP is ready and accepts data for transmission
14 HPE	HBDP Error. 0 No HBDP error 1 Error condition exists on HBDP
13 MPE	Trace memory bus Error. 0 No Trace memory bus error 1 Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow. 0 No Trace memory bus overflow 1 Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved
10 TOV	Tracebuffer Overflow. 0 No tracebuffer overflow 1 The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer (last valid write + 1)

73.4.2 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters are determine which port receives a message, based on the message source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results. See [Section 73.5.6, Output arbitration](#), for details on how the NAR arbitrates between multiple output ports. These registers should only be set in the BD NAR because partitioning is not supported in the PD NAR.

73.4.2.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Figure 1748. Source filter register (NAR_SFR)

Register index: 2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFI	FE0	SF0				0	0	0	FE1	SF1				0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	SF2				0	0	0	FE3	SF3				0	ACC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SF_n), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 1619. NAR_SFR field descriptions

Field	Description
31 SFI	Source Filter Invert. 0 Match if message source equals any of the enabled filters 1 Match if message source equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. SF0 ignored 1 Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field. See Section 73.5.6, Output arbitration for an example.
25–23	Reserved.
22 FE1	Enable Filter 1. 0 Filter 1 disabled. SF1 ignored 1 Filter 1 enabled and contained in SF1
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field. See Section 73.5.6, Output arbitration for an example.
17–15	Reserved
14 FE2	Enable Filter 2. 0 Filter 2 disabled. SF2 ignored 1 Filter 2 enabled and contained in SF2

Table 1619. NAR_SFR field descriptions (continued)

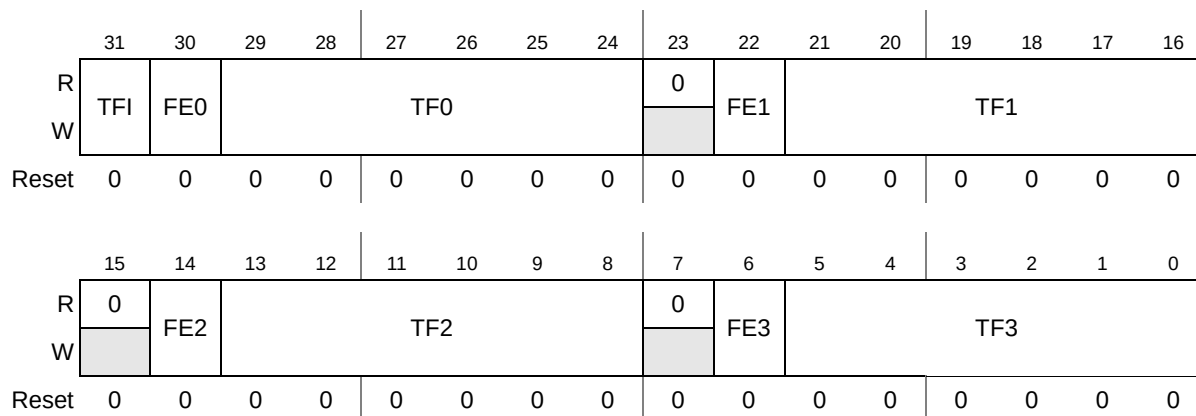
Field	Description
13–10 SF2	Source Filter 2. The message source trace ID is compared with this field. See Section 73.5.6, Output arbitration for an example.
9–7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. SF3 ignored 1 Filter 3 enabled and contained in SF3
5–2 SF3	Source Filter 3. The message source trace ID is compared with this field. See Section 73.5.6, Output arbitration for an example.
1	Reserved
0 ACC	OR source and type filters. 0 Match if source AND type filters match 1 Match if source OR type filters match

73.4.2.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Figure 1749. Message type filter register (NAR_TFR)

Register index: 3



The NAR_TFR is described in the following table.

Table 1620. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field. See Section 73.5.6, Output arbitration for an example.
23	Reserved
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field. See Section 73.5.6, Output arbitration for an example.
15	Reserved
16 FE2	Enable Filter 2. 0 Filter 2 disabled. TF2 ignored 1 Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field. See Section 73.5.6, Output arbitration for an example.
7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. TF3 ignored 1 Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field. See Section 73.5.6, Output arbitration for an example.

73.4.3 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

73.4.3.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can be transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following figures show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Figure 1750. Trace memory base address register NAR_TBAHI

Register index: 5

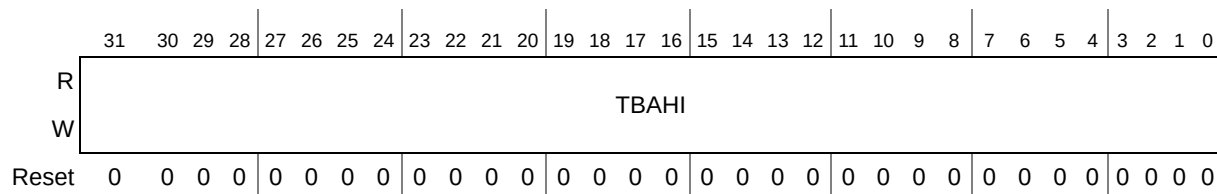
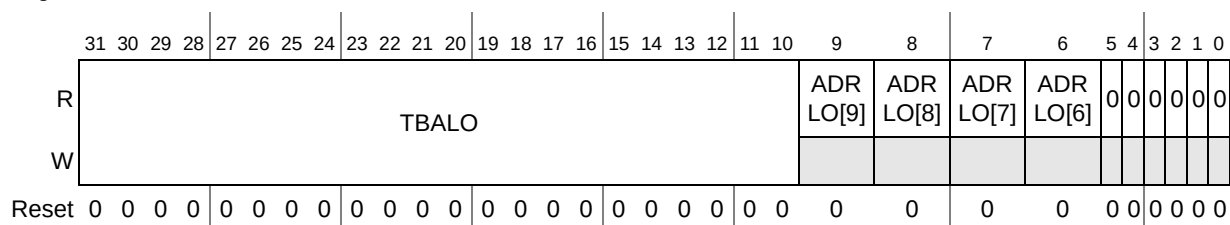


Figure 1751. Trace memory base address register NAR_TBALO

Register index: 4



The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 1621. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 1622. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)

Table 1622. NAR_TBALO field descriptions (continued)

Field	Description
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

73.4.3.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following figure shows the format of the NAR_TCR.

Figure 1752. Trace memory bus control register (NAR_TCR)

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BTM	BTS	BTH	0	0	0	TBW	MXFR								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MXFR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TCR is described in the following table.

Table 1623. NAR_TCR field description

Field	Description
31 BTM	Block Transfer Mode. 0 Data for the Trace Memory bus port is transmitted one word at a time, upon first availability 1 Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29 BTS	Block Transfer Size. 00 Block transfer size = 64 bytes 01 Block transfer size = 128 bytes 10 Block transfer size = 256 bytes 11 Block transfer size = 384 bytes

Table 1623. NAR_TCR field description (continued)

Field	Description
28–27 BTH	Block Transfer Threshold. 00 Full threshold. Number of entries in partition must equal block transfer size before block transfer starts 01 Begin transfer when block is 25% full 10 Begin transfer when block is 50% full 11 Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.
32 TBW	Trace Memory Bus Block Wrap. 0 NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached 1 NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0 MXFR	Trace Memory Bus Max Transfer Size. Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.

73.4.4 Suppress mode triggers register (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in [Section 73.5.4, Suppress mode](#). The suppress mode triggers are only available on the PD NAR and do not apply to the BD NAR. The registers are present in the BD NAR, but input suppress triggers are not connected. Each trigger can be programmed to suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each Suppress Trigger Control Register, shown in the following figure, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Figure 1753. Suppress trigger control registers (NAR_STCR1–4)

Register index: 7–A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LST															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 1624. NAR_STCR field descriptions

Field	Description
31 GST	Global Suppress Trigger. 0 The associated trigger cannot cause NAR to enter global suppress mode 1 The associated trigger is enabled for global suppress mode
29–16	Reserved
15–0 LST	Local Suppress Trigger. 0000000000000000The associated trigger is disabled for local suppress mode for all clients xxxxxxxxxxxxx1Trigger is enabled for Client 1 local suppress xxxxxxxxxxxxx1xTrigger is enabled for Client 2 local suppress xxxxxxxxxxxxx1xxTrigger is enabled for Client 3 local suppress xxxxxxxxxxxxx1xxxTrigger is enabled for Client 4 local suppress xxxxxxxxxxxxx1xxxxTrigger is enabled for Client 5 local suppress xxxxxxxxxxxxx1xxxxxTrigger is enabled for Client 6 local suppress xxxxxxxxxxxxx1xxxxxxTrigger is enabled for Client 7 local suppress xxxxxxxxxxxxx1xxxxxxxTrigger is enabled for Client 8 local suppress xxxxxxxxxxxxx1xxxxxxxTrigger is enabled for Client 9 local suppress xxxxxx1xxxxxxxxxTrigger is enabled for Client 10 local suppress xxxxx1xxxxxxxxxxTrigger is enabled for Client 11 local suppress xxxx1xxxxxxxxxxxTrigger is enabled for Client 12 local suppress xxx1xxxxxxxxxxxxTrigger is enabled for Client 13 local suppress xx1xxxxxxxxxxxxxTrigger is enabled for Client 14 local suppress x1xxxxxxxxxxxxxxTrigger is enabled for Client 15 local suppress 1xxxxxxxxxxxxxxxTrigger is enabled for Client 16 local suppress All other optionsReserved

73.4.4.1 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status (NAR_CSSR), shown in the following figure, can then be read to determine which clients are suppressed, and which are active.

Figure 1754. Client suppress status register (NAR_CSSR)

Register index: B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	LSS																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

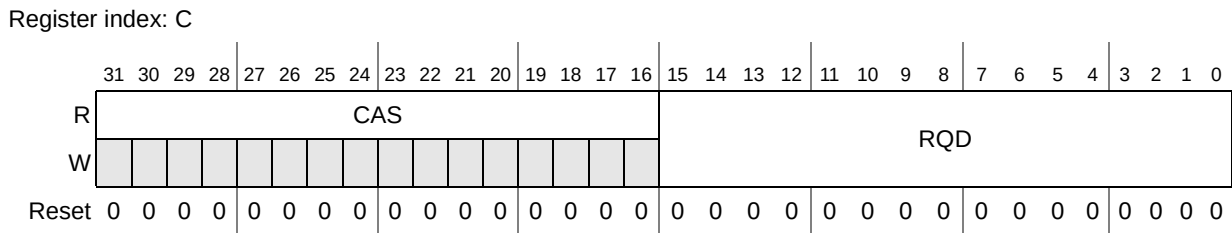
Table 1625. NAR_CCSR field descriptions

Field	Description
31–16 LSS	Local Suppress Status. 0000000000000000No client is in local suppress mode xxxxxxxxxxxxxxx1Client 1 is suppressed xxxxxxxxxxxxxxx1xClient 2 is suppressed xxxxxxxxxxxxxxx1xxClient 3 is suppressed xxxxxxxxxxxxxxx1xxxClient 4 is suppressed xxxxxxxxxxxxxxx1xxxxClient 5 is suppressed xxxxxxxxxxxxxxx1xxxxxClient 6 is suppressed xxxxxxxxxxxxxxx1xxxxxxClient 7 is suppressed xxxxxxxx1xxxxxxClient 8 is suppressed xxxxxxx1xxxxxxClient 9 is suppressed xxxxxx1xxxxxxClient 10 is suppressed xxxxx1xxxxxxClient 11 is suppressed xxxx1xxxxxxClient 12 is suppressed xxx1xxxxxxClient 13 is suppressed xx1xxxxxxClient 14 is suppressed x1xxxxxxClient 15 is suppressed 1xxxxxxClient 16 is suppressed All other optionsReserved
15–0	Reserved

73.4.5 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following figure, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.

Figure 1755. Client disable register (NAR_CDR)



The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 1626. NAR_CDR field descriptions

Field	Description
31–16 CAS	Client Activity Status. xxxxxxxxxxxxxxxx1 Client 1 is active with new data xxxxxxxxxxxxxxxx1x Client 2 is active with new data xxxxxxxxxxxxxxxx1xx Client 3 is active with new data xxxxxxxxxxxxxxxx1xxx Client 4 is active with new data xxxxxxxxxxxxxxxx1xxxx Client 5 is active with new data xxxxxxxxxxxxxxxx1xxxxx Client 6 is active with new data xxxxxxxxxxxxxxxx1xxxxxx Client 7 is active with new data xxxxxxxxxxxxxxxx1xxxxxxx Client 8 is active with new data xxxxxxxxxxxxxxxx1xxxxxxx Client 9 is active with new data xxxxxxxxxxxxxxxx1xxxxxxx Client 10 is active with new data xxxxxxxxxxxxxxxx1xxxxxxx Client 11 is active with new data xxxxx1xxxxxxxxxxxx Client 12 is active with new data xxx1xxxxxxxxxxxxxx Client 13 is active with new data xx1xxxxxxxxxxxxxxx Client 14 is active with new data x1xxxxxxxxxxxxxxx Client 15 is active with new data 1xxxxxxxxxxxxxxx Client 16 is active with new data All other optionsReserved
15–0 RQD	Receive Queue Disable. xxxxxxxxxxxxxxxx1 Disable receive queue for Client 1 xxxxxxxxxxxxxxxx1x Disable receive queue for Client 2 xxxxxxxxxxxxxxxx1xx Disable receive queue for Client 3 xxxxxxxxxxxxxxxx1xxx Disable receive queue for Client 4 xxxxxxxxxxxxxxxx1xxxx Disable receive queue for Client 5 xxxxxxxxxxxxxxxx1xxxxx Disable receive queue for Client 6 xxxxxxxxxxxxxxxx1xxxxxx Disable receive queue for Client 7 xxxxxxxxxxxxxxxx1xxxxxxx Disable receive queue for Client 8 xxxxxxxxxxxxxxxx1xxxxxxx Disable receive queue for Client 9 xxxxxxxxxxxxxxxx1xxxxxxx Disable receive queue for Client 10 xxxxxxxxxxxxxxxx1xxxxxxx Disable receive queue for Client 11 xxxxx1xxxxxxxxxxxx Disable receive queue for Client 12 xxx1xxxxxxxxxxxxxx Disable receive queue for Client 13 xx1xxxxxxxxxxxxxxx Disable receive queue for Client 14 x1xxxxxxxxxxxxxxx Disable receive queue for Client 15 1xxxxxxxxxxxxxxx Disable receive queue for Client 16 All other optionsReserved

73.4.6 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the partition related configuration for main queue (internal central queue). The following figure shows the format of the NAR_AHFPAR.

Figure 1756. Trace memory fill level and partition configuration register (NAR_AHFPAR)

Register index: E

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	AHB01kEN	0	0	0	0	0	0	AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_AHFPAR is described in the following table.

Table 1627. NAR_AHFPAR field description

Field	Description
31-11	Reserved and must be written with 0s
10 AHB01kEN	Enable 1K Marker Event
9-4	Reserved and must be written with 0s
3-0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Section 73.5.12 <i>Note: This is used in PD as it has memory ranges from 1K to 15K</i>

73.5 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

73.5.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state. This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready

to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)

3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/filtering when two output ports are enabled, enabling timestamp and so on.
4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note: *The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.*

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see [Section 73.5.4, Suppress mode](#). Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

73.5.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in this table.

Table 1628. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

73.5.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

73.5.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

73.5.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

73.5.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

73.5.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in this table on the incoming messages before transmitting to one of the output ports.

Table 1629. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8×32 -bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8×32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

73.5.3 Data reception and arbitration

The NAR supports 7 clients. Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat). The number of beats varies from client to client. The number of beats (B_n) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There are two types of clients supported at NAR client interface. The client types have their own protocols to send Nexus messages. The only common parameter is the Nexus word size (beat). One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits. Internally, the NAR converts all client interfaces to a common interface. To achieve this, the NAR instantiates interface level gaskets that handle the protocol conversion and clock domain transfer.

When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire $B_n \times 32$ data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment. Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

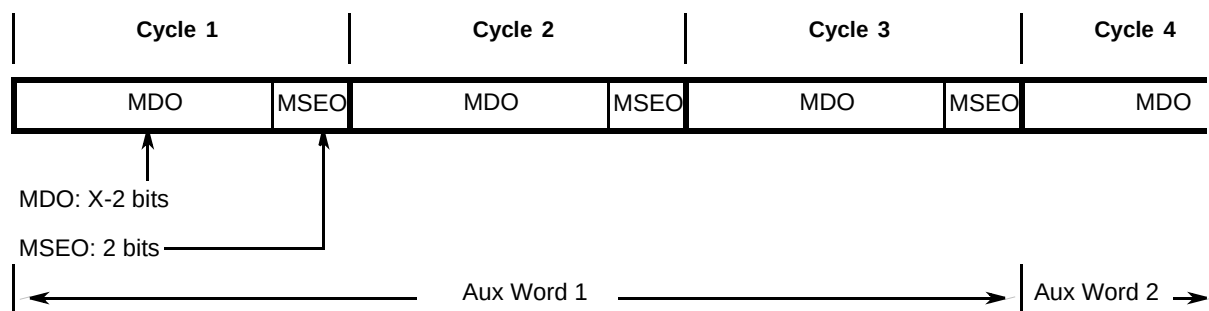


Figure 1757. AUX bus parsing

Each AUX word consists of B_n cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

73.5.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. Suppress mode is only supported in the PD NAR, it is not supported in the BD NAR. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of

message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to one set of clients, while suppress trigger two can be assigned to another set of clients, and different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

73.5.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue.

The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

73.5.6 Output arbitration

The NAR allows two output ports to be active at the same time. This feature is only available on the BD NAR. Typically, this feature is only useful if one of the ports is used to service messages of a specific type and/or messages from a particular client, while the other port supports all the rest of the message traffic. For example, the NAR could be set up to send in-circuit trace (ICT) traffic out the HBDP and to reserve the trace memory bus port for all other Nexus messages.

If two output ports are enabled, one of them must be designated as the Message Select Port (MSP). The MSP is the destination for those messages that satisfy a set of user-defined requirements as to the source and type of message. All other traffic is sent to the other enabled port. Ports are enabled by setting the port enable bits in the BD NAR_CR. The MSP is designated by setting bits [10–9] and ALT is designated by setting bits [7–6]. Once the MSP has been specified, other bits in the NAR_CR, along with the Source Filter Register (NAR_SFR) and Type Filter Register (NAR_TFR), are used to determine which messages go to the MSP and which do not. These facilities can be used to build a logical expression that can filter messages based on up to four types and/or four sources.

The NAR_SFR can be programmed with up to four source codes, while the NAR_TFR can hold up to four message-type codes. Each filter also includes an enable bit to turn it on or off. A source match occurs if a messages source code matches one of the source filters or (if the SFI bit is set) if it does not match any of the source filters. Similarly, a type match occurs if a messages type code matches one of the type filters or (if the TFI bit is set) if it does not match any of the type filters. Finally, the ACC bit of the NAR_SFR determines when both source and type filters are used, whether the two are ANDed together (ACC = 0) or ORed together (ACC = 1). For example, to send ICT messages out the HBDP and all other traffic to the trace memory bus port:

- Set ALT = 10 in the NAR_CR—sets trace memory bus port as the alternate port
- Set MSP = 01 in the NAR_CR—sets HBDP as the message select port
- Set FE0 = 1 & TF0 = 100010 (0x22) in the NAR_TFR—enables type filter 0 to match ICT messages

To send everything but data trace from source 13 to the trace memory bus port, while all other traffic goes to the HBDP:

- Set ALT = 01 in the NAR_CR—sets HBDP as the alternate port
- Set MSP = 10 in the NAR_CR—sets trace memory bus port as the message select port
- Set TFI = 1—inverts type-filter logic
- Set FE0 = 1 & TF0 = 000101 (0x05) in the NAR_TFR to select data trace of size 8 bits
- Set FE1 = 1 & TF1 = 000110 (0x06) in the NAR_TFR to select data trace of size 16 bits
- Set FE2 = 1 & TF2 = 001101 (0x0d) in the NAR_TFR to select data trace of size 32 bits
- Set FE3 = 1 & TF3 = 001110 (0x0e) in the NAR_TFR to select data trace size of 64 bits
- Set FE0 = 1 & SF0 = 001101 (0x0d) in the NAR_SFR to select source trace ID = 13
- Set ACC = 0 in NAR_SFR so the final filter is $\sim(\text{data_trace}) \text{ AND } (\text{source} = 13)\text{HBDP}$

Data is written into the output queue at a maximum rate of one entry per cycle. If the current messages source and type pass the filters, then the message is marked for the MSP. When the entry is later pulled from the queue, it is then sent to the MSP. Unmarked entries are sent to the alternate port. Message destination must be recalculated on every end-of-message boundary, using the fixed-width SRC and TYPE fields at the beginning of each Nexus message.

73.5.7 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

73.5.7.1 Aurora format

Nexus messages are pre-translated before sending to the HBDP using the Aurora protocol. The NAR output is a stream of 16×4 -bit words (where 4 is the width of the NAR Aurora out port) that are further encoded by an external Aurora block before being sent for transmission on the Aurora (HBDP) port.

Entries from the output queue are chopped into chunks of 4×16 -bit words. The amount of data that can be transmitted per Aurora cycle depends on the number of lanes available in the HBDP (transmitted HBDP word = $16 \times \text{number of lanes}$).

73.5.7.2 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and resizing in the likely case where the trace memory bus width does not equal the queue entry size.

73.5.8 Aurora interface

The external Nexus Aurora Link (NAL) contains all of the link-layer, PCS, and control logic for the HBDP channel. The Aurora data is transmitted from the NAR to the NAL for sending to the HBDP. Typically, the NAL powers up coming out of reset, but it can also be enabled by the NAR by turning on the NAR with the HBDP enabled. The reverse is true as well. If the NAL is active while the NAR is disabled, then the NAR powers up to communication-disabled mode.

Data flow between the NAR and the NAL uses a simple, synchronous interface. If the NAL is active, it indicates this by a ready signal from the NAL. If Aurora is configured as output it is indicated by a signal from the NAR.

The Aurora interface can either be operated in framing or streaming mode. Streaming mode provides the highest performance and lowest communication overhead through the link. Whereas framing mode provides greater robustness and improved error-handling at the receive side of the link. When framing mode is enabled, the NAR encapsulates the Aurora data in frames consisting of 256 Aurora chunks; that is, after every 256 Aurora transmissions, the NAR asserts end of frame and start of frame to mark the end of the last frame and the beginning of the next frame. In either mode, the NAR always asserts a start-of-frame when beginning transmission, and an end-of-frame when its output queue empties, to notify the channel partner of the end of valid data.

73.5.9 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not available. The trace memory bus access are generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (i.e. unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)
- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be $0x100 \times 0x40 = 0x4000$ bytes. The dedicated debug address range would then be $0x0D00_0000 - 0x0D00_3FFF$.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

73.5.10 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- **Device_ID**—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the

user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.

- Watchpoint—The NAR generates its own watchpoint match message under the following circumstances as shown in [Table 1633](#).
- NAR_Error (TCODE = 8)—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
 - HBDP error
 - Illegal configuration
 - Stall Error
 - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 1630. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message ⁽¹⁾	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC ⁽²⁾	NAR source ID
	4	4	fixed	ETYPE	See Table 1631
	12	12	fixed	ECODE	See Table 1632
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message ⁽³⁾	6	6	fixed	TCODE	Value = 15
	6	6	fixed	SRC ⁽²⁾	NAR source ID
	6	6	fixed	WPHIT	See Table 1633
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2011 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2011 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

73.5.10.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR

reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in this table.

Table 1631. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see Table 1632 .
0011 (h'3)	Trace memory bus port read/write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes. ⁽¹⁾
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See Table 1632	The specific configuration error is conveyed in bits 0–3 of the ECODE field (Table 1632), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.
1111 (h'F)	Stall error	Minor	—	—

1. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in this table.

Table 1632. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.
00001xxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering and/or queue partitioning disabled	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

73.5.10.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

Table 1633. Watchpoint Hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.

Table 1633. Watchpoint Hit values (continued)

WPHIT	Description	Comment
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

73.5.11 Timestamp function

The NAR and all of its clients are required to transmit timestamps, to allow precise ordering of events. All clients synchronous or asynchronous uses NAR timer value broadcasted over a bus, this ensures that all timestamps are in sync as shown in [Figure 1758](#). The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

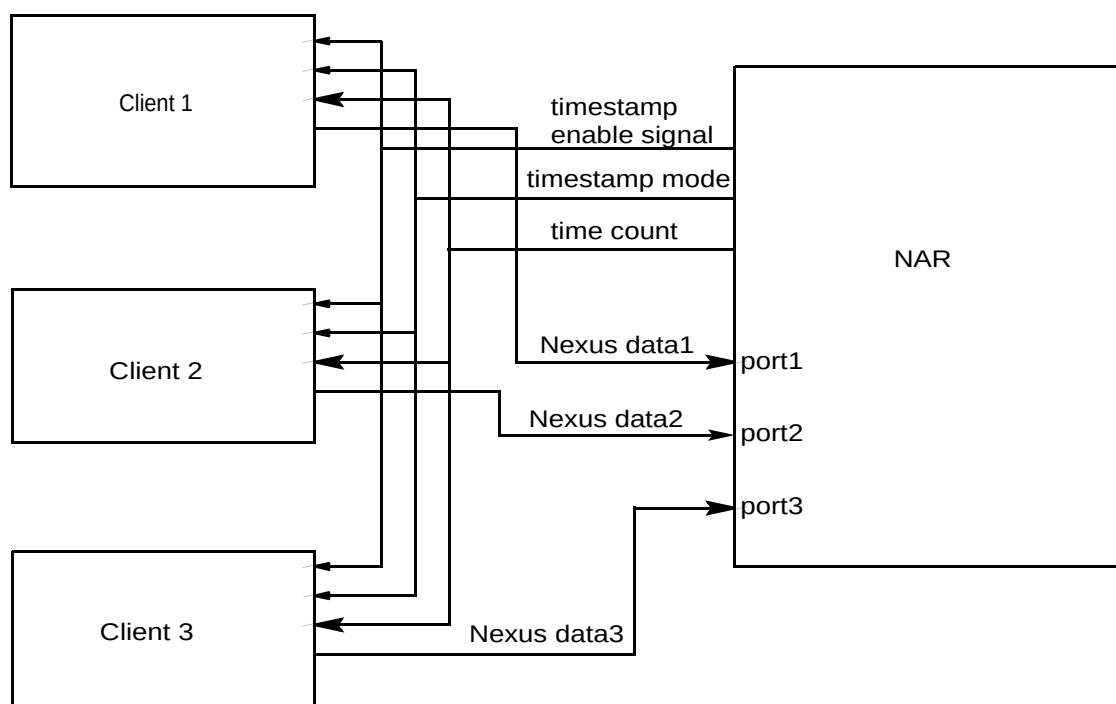


Figure 1758. Timestamp architecture

Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC

uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The Time stamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

73.5.12 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:
 - AHB start (base) address—Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)
 - Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
 - Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)
- Thus, the total block size is $(0xC0 \times 0x40) = 0x3000$ bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
 - Number of 1K blocks in 0x3000 bytes = $(3000 \text{ right shift by } 10 \text{ bits}) = C = 12$
 - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- $\text{NAR_AHFPAR}[3:0] = (\text{NAR_TBALO}[13:10] + 0xB) = (0x7 + 0xB) \text{ } 0x2$ (upper bits truncated due to width) and enable this with $\text{NAR_AHFPAR}[10] = 1$.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

73.5.13 Configuration/debug interface JTAG (1149.1)

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in [Table 1634](#). Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

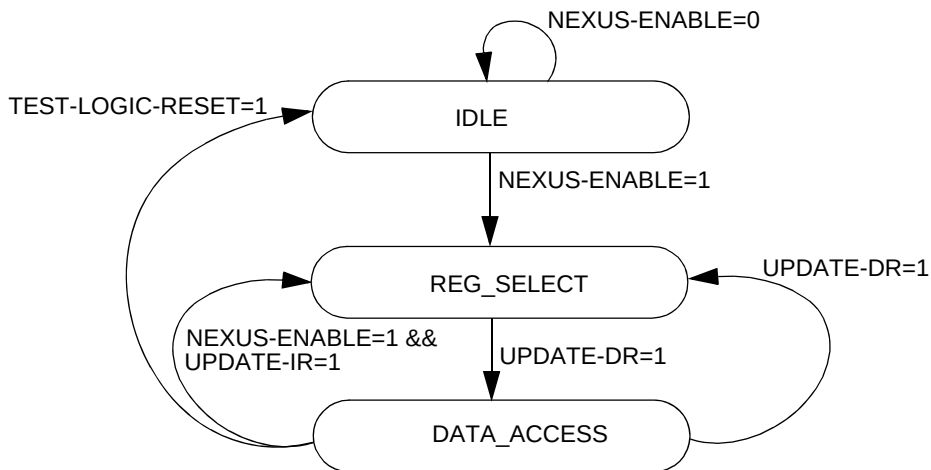


Figure 1759. Nexus controller state machine

Table 1634. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

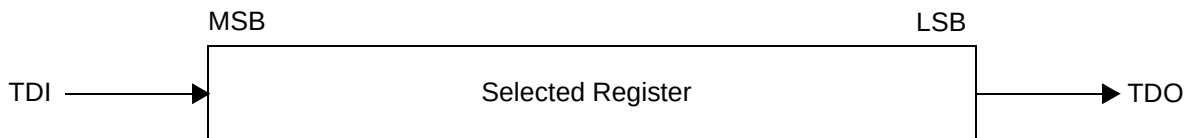


Figure 1760. Shifting data into register

73.5.13.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

73.5.13.2 Loading NEXUS-ENABLE instruction

Access to the NAR registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-

IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in [Figure 1759](#), transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 1635. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS-ENABLE instruction.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

73.5.13.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in [Figure 1761](#). The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in [Table 1616](#).

MSB	LSB
7-bit register index	R/W

Figure 1761. Nexus command format for register selection

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

This table illustrates a sequence which writes a 32-bit value to a register.

Table 1636. Writing a 32-bit value to a register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

73.5.14 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface
 - Early Write (for higher BW in special module)
 - Configurable 32/64-bit address bus width
 - 64-bit data bus width
 - Transfer Types
 - IDLE, NONSEQ, SEQ
 - Single Transfers
 - Burst Transfers
 - Configurable support for INCR
 - Configurable support for INCR4, INCR8, INCR16, WRAP4
 - Response Type
 - OKAY, ERROR
 - Transfer Size
 - Byte, Half Word, Word, Double-Word
 - Protection Control
 - Cacheable/ Non Cacheable Access Control
 - User/Privilege Access
- Endianess Support
 - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
 - Total of two transactions (One plus one ongoing)

73.5.14.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

73.5.14.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes have been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

73.5.14.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64-bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

73.5.15 Legacy client interface controller

This block converts the legacy client protocol into internal client protocol that is accepted by the internal NPC block. The Nexus beat size must be same for both interfaces. The Nexus beat is composed of MDO bits along with two MSEO bits. The trace data width from the legacy client is fixed at one beat. The architecture of this block is shown in [Figure 1762](#).

The legacy client sends each message after arbitrating the bus. It requests the gasket with a 2-bit request signal whenever it has data to send. A 00 means no request, 01 a low-priority request, 10 a medium-priority request, and 11 a high-priority request. The legacy client can send one maximum message sized data on receiving a grant from the block. There is no backpressure mechanism during transfer of a message from the client to this block. The block asserts a grant if it has at least one max message size space in its FIFO. Seeing the grant the client asserts busy signal and holds it until it wants to send data, it may hold it up to max message size data is transferred.

The following architecture describes the legacy client to internal client protocol conversion block. There is a small FIFO of parameterizable depth used to store the data from client. The block asserts a grant only when at least one max message sized space is available in FIFO. The write happens in the client's clock domain. The write pointer is passed to other domain which is NPC clock domain.

The logic on other side compares read and write pointer to check the data availability in the block's FIFO and space availability in NPC's receive queue. If space and data both are available it tries to send message to the NPC.

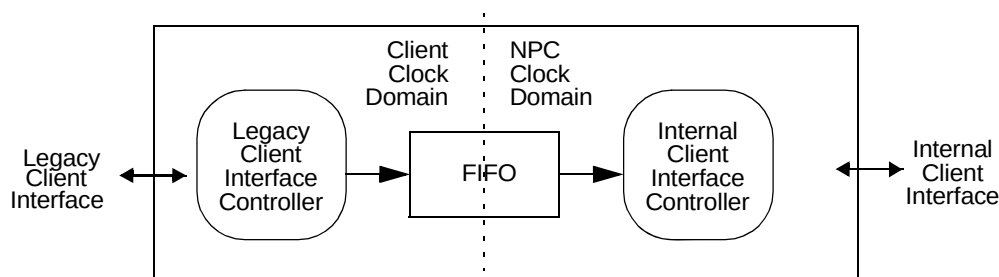


Figure 1762. Legacy client Interface controller

The size of the receive queue of the corresponding client at NPC must be known to the block. The block drives a not empty signal whenever it has data in FIFO and looks for a ready signal. It can send as many Nexus beats as receive queue can store. It operates on decrements a counter after sending data to the NPC so that it can track how much data it can send. The counter is incremented by the value of acknowledge from NPC which tells how many spaces are freed in receive queue.

73.6 Initialization information

The NAR is initialized by asserting the block reset signal for a minimum of one cycle of the slowest clock. Reset clears all latches to 0, with the only exception being the output queue, whose initialization state depends on the reset behavior of the compiled memory used. Upon exiting reset, the NAR is in the disabled state, as described in [Section 73.5.2.1, Reset](#).

Pointers and state machines are initialized by the starting of NAR functional clocks (which occurs either upon the setting of the NAR enable bit or upon the assertion of the Phy enable signal) or by an NAR soft reset.

73.7 Application information

The base application for the NAR is collecting and collating Nexus trace data. Programming examples for how to set up the NAR for trace are given in [Section 73.5.6, Output arbitration](#).

74 e200z420n3 Nexus 3 Module

The e200z420n3 Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port that is used in conjunction with the JTAG port in these processors.

74.1 Introduction

74.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

74.1.2 Terms and Definitions

[Table 1637](#) contains a set of terms and definitions associated with the Nexus 3 module.

Table 1637. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.

Table 1637. Terms and Definitions (continued)

Term	Description
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	“System-on-a-Chip”. SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase “according to the standard” is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

74.1.3 Feature List

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated.

An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.

- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging
- Auxiliary interface for higher data input/output
 - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
 - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
 - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
 - One (1) Read/Write Error pin (**nex_err_b**) pin
 - One (1) Watchpoint Event output pin (**nex_evto_b**)
 - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
 - One (1) Event In pin (**nex_evti_b**)
 - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)

Note: For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM — maximum number of MDO pins) or Reduced Port Mode (RPM — minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note: The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

74.1.4 Functional Block Diagram

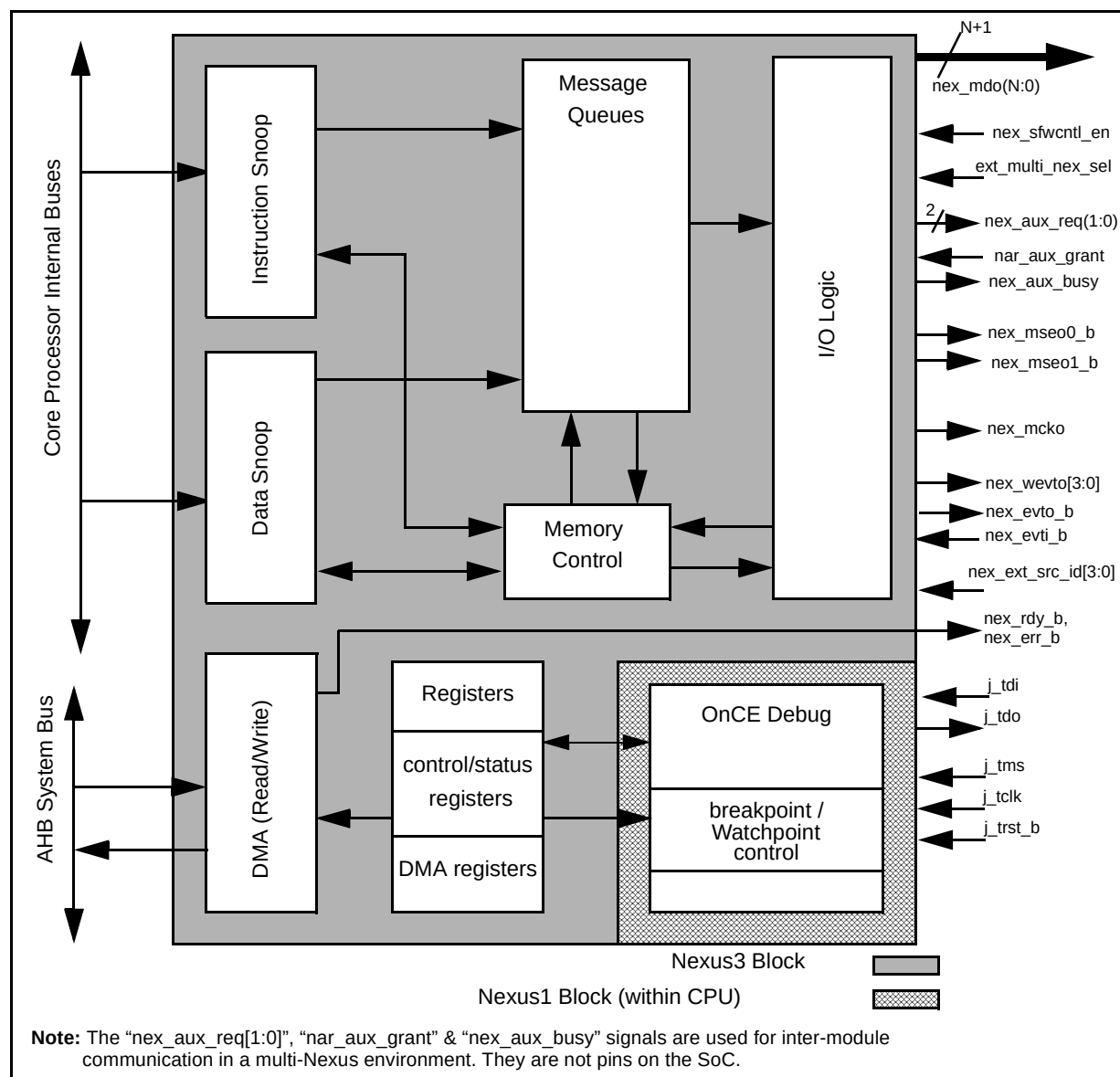


Figure 1763. Nexus 3 functional block diagram

74.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (NEXUS3-ACCESS) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the

Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test-Logic-Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

74.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

74.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in [Table 1638](#).

Table 1638. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace — Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace — Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
Data Trace — Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 1643)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace — Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 1643)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)

Table 1638. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from DEVENT _{DQTAG} register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace — Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace — Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace — Indirect Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Data Trace — Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 1643)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table 1638. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Trace — Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 1643)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 1641) — indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section 74.12.4, Resource Full Messages.)
Program Trace — Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	1	32	HIST	variable	branch / predicate instruction history (see Section 74.12.1, Branch Trace Messaging types)
Program Trace — Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Section 74.12.1, Branch Trace Messaging types.)

Table 1638. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Program Trace — Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 1642)
	2	2	CDF	fixed	# fields of information in CDATE. 00 — reserved, 01 — one field (CDATA1) (reserved), 10 — two fields (CDATA1 + CDATE2), 11 — three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 — [branch / predicate instruction history] (See Section 74.12.5, Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2 — PID info (See Section 74.12.5, Program Correlation Messages.)

Note: Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in [Section 74.12.1, Branch Trace Messaging types](#). If the Branch History method is selected, the shaded TCODEs above will not be messaged out.

[Table 1639](#) shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 1639. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

[Table 1640](#) shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 1640. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 – 1111	Reserved

[Table 1641](#) shows the encodings used for resource codes for certain messages.

Table 1641. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

[Table 1642](#) shows the event code encodings used for certain messages.

Table 1642. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010–0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via mtspr PID0/NPIDR
0110–1000	Reserved for future functionality
1001	Begin masking of program trace messages due to $MSR_{PMM} = 0$ and $DC4_{PTMARK} = 1$
1010	Branch and link occurrence (direct branch function call)
1011–1111	Reserved for future functionality

[Table 1643](#) shows the data trace size encodings used for certain messages.

Table 1643. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 — no data
0001	Byte

Table 1643. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14) (continued)

DTM Size Encoding	Transfer Size
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001–1111	Reserved

74.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#) and [Section 74.6, Nexus 3 Register Access via Software](#) for details on Nexus 3 register access.

Note: Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

[Table 1644](#) details the register map for the Nexus 3 module.

Table 1644. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/Write	Read Address	Write Address	DCR # ⁽¹⁾
Client Select Control (CSC) ⁽²⁾	0x1	R	0x02	—	
Port Configuration Register (PCR) ⁽³⁾	PCR_INDEX ²	R/W	—	—	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	—
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	—
Reserved	0x8	R/W	0x18	0x19	—
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	—
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	—
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	—
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377

Table 1644. Nexus 3 Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/Write	Read Address	Write Address	DCR # ⁽¹⁾
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 → 0x2F	—	0x28→0x5E	0x29→0x5F	—
Development Status (DS)	0x30	R	0x60	—	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WSMK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	—	0x68	0x69	—
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 → 0x3F	—	0x72→0x7E	0x73→0x7F	—

1. Software access via the **mfdcr** and **mtocr** instructions use these values for the DCR number. Software writes to these registers via **mtocr** when **nex_sfvcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

74.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.

Figure 1764. Client Select Control (CSC) register

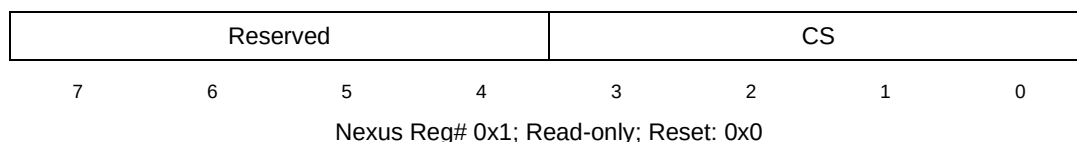


Table 1645. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx Nexus client (SoC level)

74.4.2 Port Configuration Register (PCR) — reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.

Figure 1765. Port Configuration Register (PCR)

OPC	0	MCK_EN	MCK_DIV	0																											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# PCR_INDEX; Read/Write; Reset: 0x0

Table 1646. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# nex_mdo[n:0] pins defined by SOC) 1 Full Port Mode configuration (max# nex_mdo[n:0] pins defined by SOC)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 nex_mcko is disabled 1 nex_mcko is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 nex_mcko is 1x processor clock freq. 001 nex_mcko is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 nex_mcko is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 nex_mcko is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note: The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

74.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in [Figure 1766](#) and its fields are described in [Table 1647](#).

Figure 1766. Development Control 1 (DC1) register

OPC	MCK_DIV	0	PTM	0	OTS	0	POTD	TSEN	EOC	EIC	0	TM																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x2; DCR: 368; Read/Write; Reset: 0x0

Table 1647. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 — Reduced Port Mode configuration (min# nex_mdo[n:0] pins defined) 1 — Full Port Mode configuration (max# nex_mdo[n:0] pins defined)
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 — nex_mcko is 1x processor clock freq. 01 — nex_mcko is 1/2x processor clock freq. 10 — nex_mcko is 1/4x processor clock freq. 11 — nex_mcko is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 — Program Trace uses traditional Branch Messages 1 — Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 — PID0 data is transmitted within Ownership Trace Messages 1 — Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 — Periodic Ownership Trace message events are enabled 1 — Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable — (not implemented, write to 00) 00 — Timestamp is disabled
11:10	EOC	EVTO Control 00 — nex_evto_b upon occurrence of Watchpoints (configured in DC2 and DC3) 01 — nex_evto_b upon entry into Debug Mode 1x — Reserved

Table 1647. DC1 field descriptions (continued)

Bits	Name	Description
9:8	EIC	EVTI Control 00 — nex_evti_b is used for synchronization (Program Trace Sync msg is generated) 01 — nex_evti_b is used for Debug request 10 — nex_evti_b is disabled 1x — Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode ⁽¹⁾ 000000 — All Trace Disabled xxxxx1 — Ownership Trace enabled xxxx1x — Data Trace enabled xxx1xx — Program Trace enabled xx1xxx — Watchpoint Trace enabled x1xxxx — Reserved 1xxxxx — Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to [Section 74.4.7, Watchpoint Trigger \(WT, PTSTC, PTETC, DTSTC, DTETC\) registers](#) for more information on watchpoint triggering. Refer to [Section 74.16, External Hardware Trigger Controls](#) for more information on external hardware triggering.

Note: The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) **MUST ONLY** be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

74.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z420n3) Core Debug Support chapter.

Development Control Register 2 is shown in [Figure 1767](#) and its fields are described in [Table 1648](#).

Figure 1767. Development Control 2 (DC2) register

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	EWC
-----------	-----------	-----------	-----------	-----

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x3; DCR: 369; Read/Write; Reset: 0x0

Table 1648. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 — No Watchpoints #0–14 trigger nex_wevto[3] 0001 — Watchpoint #0 triggers nex_wevto[3] 0010 — Watchpoint #1 triggers nex_wevto[3] 0011 — Watchpoint #2 triggers nex_wevto[3] 0100 — Watchpoint #3 triggers nex_wevto[3] 0101 — Watchpoint #4 triggers nex_wevto[3] 0110 — Watchpoint #5 triggers nex_wevto[3] 0111 — Watchpoint #6 triggers nex_wevto[3] 1000 — Watchpoint #7 triggers nex_wevto[3] 1001 — Watchpoint #8 triggers nex_wevto[3] 1010 — Watchpoint #9 triggers nex_wevto[3] 1011 — Watchpoint #10 triggers nex_wevto[3] 1100 — Watchpoint #11 triggers nex_wevto[3] 1101 — Watchpoint #12 triggers nex_wevto[3] 1110 — Watchpoint #13 triggers nex_wevto[3] 1111 — Watchpoint #14 triggers nex_wevto[3]
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 — No Watchpoints #0–14 trigger nex_wevto[2] 0001 — Watchpoint #0 triggers nex_wevto[2] 0010 — Watchpoint #1 triggers nex_wevto[2] 0011 — Watchpoint #2 triggers nex_wevto[2] 0100 — Watchpoint #3 triggers nex_wevto[2] 0101 — Watchpoint #4 triggers nex_wevto[2] 0110 — Watchpoint #5 triggers nex_wevto[2] 0111 — Watchpoint #6 triggers nex_wevto[2] 1000 — Watchpoint #7 triggers nex_wevto[2] 1001 — Watchpoint #8 triggers nex_wevto[2] 1010 — Watchpoint #9 triggers nex_wevto[2] 1011 — Watchpoint #10 triggers nex_wevto[2] 1100 — Watchpoint #11 triggers nex_wevto[2] 1101 — Watchpoint #12 triggers nex_wevto[2] 1110 — Watchpoint #13 triggers nex_wevto[2] 1111 — Watchpoint #14 triggers nex_wevto[2]
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 — No Watchpoints #0–14 trigger nex_wevto[1] 0001 — Watchpoint #0 triggers nex_wevto[1] 0010 — Watchpoint #1 triggers nex_wevto[1] 0011 — Watchpoint #2 triggers nex_wevto[1] 0100 — Watchpoint #3 triggers nex_wevto[1] 0101 — Watchpoint #4 triggers nex_wevto[1] 0110 — Watchpoint #5 triggers nex_wevto[1] 0111 — Watchpoint #6 triggers nex_wevto[1] 1000 — Watchpoint #7 triggers nex_wevto[1] 1001 — Watchpoint #8 triggers nex_wevto[1] 1010 — Watchpoint #9 triggers nex_wevto[1] 1011 — Watchpoint #10 triggers nex_wevto[1] 1100 — Watchpoint #11 triggers nex_wevto[1] 1101 — Watchpoint #12 triggers nex_wevto[1] 1110 — Watchpoint #13 triggers nex_wevto[1] 1111 — Watchpoint #14 triggers nex_wevto[1]

Table 1648. DC2 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 — No Watchpoints #0–14 trigger nex_wevto[0] 0001 — Watchpoint #0 triggers nex_wevto[0] 0010 — Watchpoint #1 triggers nex_wevto[0] 0011 — Watchpoint #2 triggers nex_wevto[0] 0100 — Watchpoint #3 triggers nex_wevto[0] 0101 — Watchpoint #4 triggers nex_wevto[0] 0110 — Watchpoint #5 triggers nex_wevto[0] 0111 — Watchpoint #6 triggers nex_wevto[0] 1000 — Watchpoint #7 triggers nex_wevto[0] 1001 — Watchpoint #8 triggers nex_wevto[0] 1010 — Watchpoint #9 triggers nex_wevto[0] 1011 — Watchpoint #10 triggers nex_wevto[0] 1100 — Watchpoint #11 triggers nex_wevto[0] 1101 — Watchpoint #12 triggers nex_wevto[0] 1110 — Watchpoint #13 triggers nex_wevto[0] 1111 — Watchpoint #14 triggers nex_wevto[0]
15:0	EWC	EVTO Watchpoint Configuration ⁽¹⁾ 0000000000000000 No Watchpoints #0–15 trigger nex_evto_b xxxxxxxxxxxxxxxx1 Watchpoint #0 triggers nex_evto_b xxxxxxxxxxxxxxxx1x Watchpoint #1 triggers nex_evto_b xxxxxxxxxxxxxxxx1xx Watchpoint #2 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxx Watchpoint #3 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxxx Watchpoint #4 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxxxx Watchpoint #5 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxxxxx Watchpoint #6 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxxxxxx Watchpoint #7 triggers nex_evto_b xxxxxxxx1xxxxxxx Watchpoint #8 triggers nex_evto_b xxxxxx1xxxxxxx Watchpoint #9 triggers nex_evto_b xxxxx1xxxxxxx Watchpoint #10 triggers nex_evto_b xxxx1xxxxxxx Watchpoint #11 triggers nex_evto_b xxx1xxxxxxx Watchpoint #12 triggers nex_evto_b xx1xxxxxxx Watchpoint #13 triggers nex_evto_b x1xxxxxxx Watchpoint #14 triggers nex_evto_b 1xxxxxxx Watchpoint #15 triggers nex_evto_b

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in [Figure 1768](#) and its fields are described in [Table 1649](#).

Figure 1768. Development Control 3 (DC3) register

WEVTO[3]C																																WEVTO[2]C																																WEVTO[1]C																																WEVTO[0]C																																0																																EWC																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																																30																																29																																28																																27																																26																																25																																24																																23																																22																																21																																20																																19																																18																																17																																16																																15																																14																																13																																12																																11																																10																																9																																8																																7																																6																																5																																4																																3																																2																																1																																0																															
Nexus Reg# 0x4; DCR: 370; Read/Write; Reset: 0x0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															

Table 1649. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #15–#26 trigger nex_wevto[3] 0001 Watchpoint #15 triggers nex_wevto[3] 0010 Watchpoint #16 triggers nex_wevto[3] 0011 Watchpoint #17 triggers nex_wevto[3] 0100 Watchpoint #18 triggers nex_wevto[3] 0101 Watchpoint #19 triggers nex_wevto[3] 0110 Watchpoint #20 triggers nex_wevto[3] 0111 Watchpoint #21 triggers nex_wevto[3] 1000 Watchpoint #22 triggers nex_wevto[3] 1001 Watchpoint #23 triggers nex_wevto[3] 1010 Watchpoint #24 triggers nex_wevto[3] 1011 Watchpoint #25 triggers nex_wevto[3] 1100 Watchpoint #26 triggers nex_wevto[3] 11x1 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #15–#26 trigger nex_wevto[2] 0001 Watchpoint #15 triggers nex_wevto[2] 0010 Watchpoint #16 triggers nex_wevto[2] 0011 Watchpoint #17 triggers nex_wevto[2] 0100 Watchpoint #18 triggers nex_wevto[2] 0101 Watchpoint #19 triggers nex_wevto[2] 0110 Watchpoint #20 triggers nex_wevto[2] 0111 Watchpoint #21 triggers nex_wevto[2] 1000 Watchpoint #22 triggers nex_wevto[2] 1001 Watchpoint #23 triggers nex_wevto[2] 1010 Watchpoint #24 triggers nex_wevto[2] 1011 Watchpoint #25 triggers nex_wevto[2] 1100 Watchpoint #26 triggers nex_wevto[2] 11x1 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #15–#29 trigger nex_wevto[1] 0001 Watchpoint #15 triggers nex_wevto[1] 0010 Watchpoint #16 triggers nex_wevto[1] 0011 Watchpoint #17 triggers nex_wevto[1] 0100 Watchpoint #18 triggers nex_wevto[1] 0101 Watchpoint #19 triggers nex_wevto[1] 0110 Watchpoint #20 triggers nex_wevto[1] 0111 Watchpoint #21 triggers nex_wevto[1] 1000 Watchpoint #22 triggers nex_wevto[1] 1001 Watchpoint #23 triggers nex_wevto[1] 1010 Watchpoint #24 triggers nex_wevto[1] 1011 Watchpoint #25 triggers nex_wevto[1] 1100 Watchpoint #26 triggers nex_wevto[1] 1101 Watchpoint #27 triggers nex_wevto[1] 1110 Watchpoint #28 triggers nex_wevto[1] 1111 Watchpoint #29 triggers nex_wevto[1]

Table 1649. DC3 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 No Watchpoints #15–#29 trigger nex_wevto[0] 0001 Watchpoint #15 triggers nex_wevto[0] 0010 Watchpoint #16 triggers nex_wevto[0] 0011 Watchpoint #17 triggers nex_wevto[0] 0100 Watchpoint #18 triggers nex_wevto[0] 0101 Watchpoint #19 triggers nex_wevto[0] 0110 Watchpoint #20 triggers nex_wevto[0] 0111 Watchpoint #21 triggers nex_wevto[0] 1000 Watchpoint #22 triggers nex_wevto[0] 1001 Watchpoint #23 triggers nex_wevto[0] 1010 Watchpoint #24 triggers nex_wevto[0] 1011 Watchpoint #25 triggers nex_wevto[0] 1100 Watchpoint #26 triggers nex_wevto[0] 1101 Watchpoint #27 triggers nex_wevto[0] 1110 Watchpoint #28 triggers nex_wevto[0] 1111 Watchpoint #29 triggers nex_wevto[0]
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration ⁽¹⁾ 0000000000000000 No Watchpoints #16–#29 trigger nex_evto_b xxxxxxxxxxxx1 Watchpoint #16 triggers nex_evto_b xxxxxxxxxxxx1x Watchpoint #17 triggers nex_evto_b xxxxxxxxxxxx1xx Watchpoint #18 triggers nex_evto_b xxxxxxxxxxxx1xxx Watchpoint #19 triggers nex_evto_b xxxxxxxxxxxx1xxxx Watchpoint #20 triggers nex_evto_b xxxxxxxx1xxxxx Watchpoint #21 triggers nex_evto_b xxxxxxxx1xxxxxx Watchpoint #22 triggers nex_evto_b xxxxxx1xxxxxxx Watchpoint #23 triggers nex_evto_b xxxxx1xxxxxxx Watchpoint #24 triggers nex_evto_b xxxx1xxxxxxx Watchpoint #25 triggers nex_evto_b xxx1xxxxxxx Watchpoint #26 triggers nex_evto_b xx1xxxxxxx Watchpoint #27 triggers nex_evto_b x1xxxxxxx Watchpoint #28 triggers nex_evto_b 1xxxxxxx Watchpoint #29 triggers nex_evto_b

1. EOC bits in DC1 must be programmed to trigger EVTO on Watchpoint occurrence for EWC bits to have any effect.

74.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in [Figure 1769](#) and its fields are described in [Table 1650](#).

Figure 1769. Development Control 4 (DC4) register

PTMARK	DTMARK	0														EVCDM															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x5; DCR: 371; Read/Write; Reset: 0x0

Table 1650. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR _{PMM} for masking program trace messages 1 Mask program trace messages when MSR _{PMM} =‘0’, unmask program trace messages when MSR _{PMM} =‘1’
30	DTMARK	Data Trace Mark 0 Ignore MSR _{PMM} for masking data trace messages 1 Mask data trace messages when MSR _{PMM} =‘0’, unmask data trace messages when MSR _{PMM} =‘1’
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask ⁽¹⁾ 0000000000000000 No EVCODEs masked for Program Correlation Messages xxxxxxxxxxxxxxxx1 EVCODE #0 is masked for Program Correlation Messages xxxxxxxxxxxxxxxx1x EVCODE #1 is masked for Program Correlation Messages xxxxxxxxxxxxxxxx1xx EVCODE #2 is masked for Program Correlation Messages xxxxxxxxxxxxxxxx1xxx EVCODE #3 is masked for Program Correlation Messages xxxxxxxxxxxx1xxxx EVCODE #4 is masked for Program Correlation Messages xxxxxxxxxxxx1xxxxx EVCODE #5 is masked for Program Correlation Messages xxxxxxxxxxxx1xxxxxx EVCODE #6 is masked for Program Correlation Messages xxxxxxxx1xxxxxxxx EVCODE #7 is masked for Program Correlation Messages xxxxxxxx1xxxxxxxxx EVCODE #8 is masked for Program Correlation Messages xxxxxx1xxxxxxxxxx EVCODE #9 is masked for Program Correlation Messages xxxxx1xxxxxxxxxxx EVCODE #10 is masked for Program Correlation Messages xxxx1xxxxxxxxxxxx EVCODE #11 is masked for Program Correlation Messages xxx1xxxxxxxxxxxxx EVCODE #12 is masked for Program Correlation Messages xx1xxxxxxxxxxxxxx EVCODE #13 is masked for Program Correlation Messages x1xxxxxxxxxxxxxxx EVCODE #14 is masked for Program Correlation Messages 1xxxxxxxxxxxxxxx EVCODE #15 is masked for Program Correlation Messages

1. Refer to [Table 1642](#) for implemented EVCODEs.

74.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.

Figure 1770. Development Status (DS) register

DBG	LPS	LPC	0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			

Nexus Reg# 0x4; DCR: 409; Read-only; Reset: 0x0

Table 1651. DS field descriptions

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (jd_debug_b signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting State (p_waiting signal asserted) 010 Halted State (p_halted signal asserted) 011 Reserved 1xx Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (p_halted signal asserted) 10 CPU in Stopped state (p_stopped signal asserted) 11 CPU in Waiting state (p_waiting signal asserted)
25:0	—	Reserved for future functionality (read as 0)

74.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related “window” for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For

example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z420n3) Core Debug Support chapter.

Figure 1771. Watchpoint Trigger (WT) register

PTS		PTE		DTS		DTE		0																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0xB; DCR: 375; Read/Write; Reset: 0x0

[Table 1652](#) details the Watchpoint Trigger register fields.

Table 1652. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

Figure 1772. Program Trace Start Trigger Control (PTSTC) register

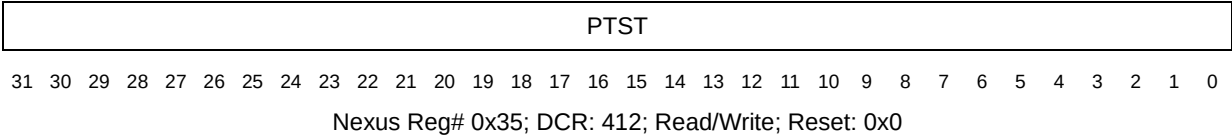


Table 1653 details the PTSTC register fields.

Table 1653. PTSTC field descriptions

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control
		00000000000000000000000000000000 Trigger disabled
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #8
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #9
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #10
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #11
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #12
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #13
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #14
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #15
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #16
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #17
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #18
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #19
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #20
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #21
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #22
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #23
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #24
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #25
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #26
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #27
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #28
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #29
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #30
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

Figure 1773. Program Trace End Trigger Control (PTETC) register

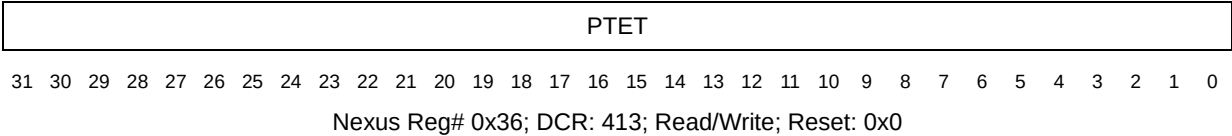


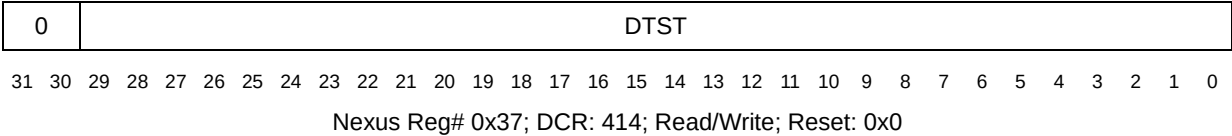
Table 1654 details the PTETC register fields.

Table 1654. PTETC field descriptions

Bits	Name	Description
31:0	PTET	Program Trace End Trigger Control
		00000000000000000000000000000000 Trigger disabled
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #8
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #9
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #10
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #11
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #12
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #13
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #14
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #15
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #16
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #17
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #18
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #19
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #20
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #21
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #22
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #23
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #24
		xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #25
		xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #26
		xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #27
		xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #28
		xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #29
		x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #30
		1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

Figure 1774. Data Trace Start Trigger Control (DTSTC) register



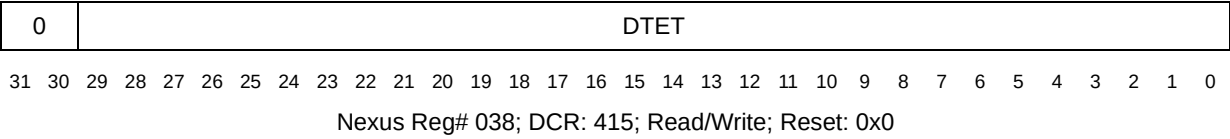
The following table details the DTSTC register fields.

Table 1655. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #16 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #17 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #18 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #19 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #20 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #21 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #22 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #23 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #24 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #25 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #26 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #27 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #28 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

Figure 1775. Data Trace End Trigger Control (DTETC) register



The following table details the DTETC register fields.

Table 1656. DTETC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #16 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #17 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #18 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #19 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #20 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #21 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #22 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #23 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #24 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #25 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #26 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #27 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #28 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #29

74.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages (DC1_{TM} must also be programmed to generate Watchpoint Trace Messages).

Figure 1776. Watchpoint Mask (WMSK) register

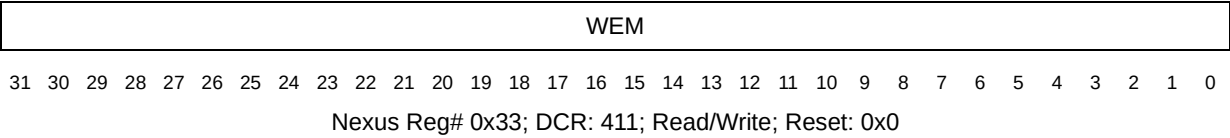


Table 1657 details the Watchpoint Trigger register fields.

Table 1657. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Watchpoint #0 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Watchpoint #1 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Watchpoint #2 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Watchpoint #3 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Watchpoint #4 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Watchpoint #5 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Watchpoint #6 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Watchpoint #7 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Watchpoint #8 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Watchpoint #9 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Watchpoint #10 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #11 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Watchpoint #12 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #13 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #14 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #15 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #16 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #17 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #18 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #19 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #20 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #21 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #22 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #23 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #24 enabled for WTM
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #25 enabled for WTM
		xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #26 enabled for WTM
		xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #27 enabled for WTM
		xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #28 enabled for WTM
		xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #29 enabled for WTM
		x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #30 enabled for WTM
		1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #31 enabled for WTM

74.4.9 Nexus Overrun Control (OVCR) register

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

Figure 1777. Nexus Overrun Control (OVCR) register

0		SPTHOLD		0				SPEN				0		STTHOLD		0								STEN							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x32; DCR: 410; Read/Write; Reset: 0x0

Figure 1778. OVCR field descriptions

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 Suppression threshold is when message queues are 1/4 full 01 Suppression threshold is when message queues are 1/2 full 10 Suppression threshold is when message queues are 3/4 full 11 Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 Suppression is disabled xxxxx1 Ownership Trace message suppression is enabled xxxx1x Data Trace message suppression is enabled xxx1xx Program Trace message suppression is enabled xx1xxx Watchpoint Trace message suppression is enabled x1xxxx Reserved 1xxxxx Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 Stall threshold is when message queues are 1/4 full 01 Stall threshold is when message queues are 1/2 full 10 Stall threshold is when message queues are 3/4 full 11 Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 Stalling is disabled 1 Stalling is enabled

74.4.10 Data Trace Control (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

Figure 1779. Data Trace Control (DTC) register

RWT1	RWT2	RWT3	RWT4	0								STDC1	STDC2	STDC3	STDC4	0				RC1	RC2	RC3	RC4	DI	0						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0xD; DCR: 376; Read/Write; Reset: 0x0

The following table details the Data Trace Control register fields.

Table 1658. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs

Table 1658. DTC field descriptions (continued)

Bits	Name	Description
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses <i>Note: The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module.</i> <i>Note: The setting of the DI bit also affects the information provided on the data trace port outputs (See the “Data Trace Port Signals” section in the Core (e200z420n3) Core Debug Support chapter.)</i>
2:0	—	Reserved for future functionality (read as 0)

74.4.11 Data Trace Start Address Registers (DTSA1–4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Figure 1780. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0xE; DCR: 377; Read/Write; Reset: 0x0																															

Figure 1781. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0xF; DCR: 378; Read/Write; Reset: 0x0																															

Figure 1782. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x10; DCR: 379; Read/Write; Reset: 0x0

Figure 1783. Data Trace Start Address 4 (DTSA4) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x11; DCR: 380; Read/Write; Reset: 0x0

74.4.12 Data Trace End Address 1–4 (DTEA1–4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Figure 1784. Data Trace End Address 1 register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x12; DCR: 381; Read/Write; Reset: 0x0

Figure 1785. Data Trace End Address 2 register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x13; DCR: 382; Read/Write; Reset: 0x0

Figure 1786. Data Trace End Address 3 register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x14; DCR: 383; Read/Write; Reset: 0x0

Figure 1787. Data Trace End Address 4 register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x15; DCR: 408; Read/Write; Reset: 0x0

[Table 1659](#) illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 1659. Data trace — address range options

Programmed values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA → ← DTEA
DTSA < DTEA	1	← DTSA DTEA →
DTSA > DTEA	N/A	Invalid Range — no trace
DTSA = DTEA	N/A	Invalid Range — no trace

Note: *DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are inclusive of the DTSA and DTEA addresses for Range Control settings indicating “within range,” and are exclusive of the DTSA and DTEA addresses for Range Control settings indicating “outside of range.”*

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when $DTC_{DI} = 1$, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

74.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes. Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values. The RWCS Register also provides Read/Write Access Status information per [Table 1661](#).

Figure 1788. Read/Write Access Control/Status (RWCS) register

AC	RW	SZ		MAP		PR		ATTR		0		CNT																ERR	DV		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x7; Read/Write⁽¹⁾; Reset: 0x0

1. ERR and DV are read-only

Table 1660. RWCS field descriptions

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access

Table 1660. RWCS field descriptions (continued)

Bits	Name	Description
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 1xx Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map All other values are reserved
RWCS[23:22]	PR ⁽¹⁾	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx p_d_hprot[4] driven to 0 for accesses x1xx p_d_hprot[4] driven to 1 for accesses xx0x p_d_hprot[3] driven to 0 for accesses xx1x p_d_hprot[3] driven to 1 for accesses xxx0 p_d_hprot[2] driven to 0 for accesses xxx1 p_d_hprot[2] driven to 1 for accesses
RWCS[17:16]	—	Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR ⁽²⁾	Read/Write Access Error (see Table 1661)
RWCS[0]	DV ⁽²⁾	Read/Write Access Data Valid (see Table 1661)

1. The priority functionality is not currently implemented.

2. ERR and DV are read-only.

Table 1661. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

74.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Figure 1789. Read/Write Access Data (RWD) register

Read/Write Data																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0xA; Read/Write; Reset: 0x0																															

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. [Table 1662](#) shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 1662. RWD data placement for transfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	—	—	—	X
Half	x x 0	0 0 1	—	—	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword						
first RWD pass (low order data)	0 0 0	0 1 1	X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

“X” indicates byte lanes with valid data

“—” indicates byte lanes that will contain unused data.

[Table 1663](#) shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 1663. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	AHB[7:0]
Byte @001	0 0 1	—	—	—	AHB[15:8]
Byte @010	0 1 0	—	—	—	AHB[23:16]
Byte @011	0 1 1	—	—	—	AHB[31:24]
Byte @100	1 0 0	—	—	—	AHB[39:32]
Byte @101	1 0 1	—	—	—	AHB[47:40]
Byte @110	1 1 0	—	—	—	AHB[55:48]
Byte @111	1 1 1	—	—	—	AHB[63:56]
Half @000	0 0 0	—	—	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	—	—	AHB[31:24]	AHB[23:16]

Table 1663. RWD byte lane mapping (continued)

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Half @100	1 0 0	—	—	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	—	—	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

“—” indicates byte lanes that will contain unused data.

74.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Figure 1790. Read/Write Access Address (RWA) register

Read/Write Address																													
--------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x9; Read/Write; Reset: 0x0

74.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction (“NEXUS3-ACCESS”) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in [Table 1644](#).

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See [Section 74.23, IEEE 1149.1 \(JTAG\) RD/WR sequences](#).)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see [Table 1644](#)), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

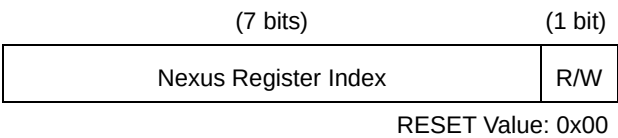


Figure 1791. JTAG DR format

Table 1664. JTAG DR field descriptions

Nexus Register Index	Selected from values in Table 1644
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
- a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the “Capture-DR” state.
 - b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the “Update-DR” state.

74.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdc** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. [Table 1644](#) shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

74.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

74.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

74.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

74.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) = 0x0003FC01, New Address (A2) = 0x0003F365

Message Generation: A1 = 0000 0000 0000 0011 1111 1100 0000 0001 A2 = 0000 0000 0000 0011 1111 0011 0110 0101 A1 $\oplus$ A2 = 0000 0000 0000 0000 0000 1111 0110 0100 Address Message (M1) = 1111 0110 0100 Address Re-creation: A1 $\oplus$ M1 = A2 A1 = 0000 0000 0000 0011 1111 1100 0000 0001 M1 = 0000 0000 0000 0000 0000 1111 0110 0100 A2 = 0000 0000 0000 0011 1111 0011 0110 0101
--

Figure 1792. Relative address generation and recreation

74.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

74.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

74.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

74.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to [Section 74.4.9, Nexus Overrun Control \(OVCR\) register](#), for complete programming options.

74.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in [Section 74.8.1, Message Queue Overrun](#). Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

74.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

[Table 1665](#) lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in [Table 1665](#).

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction 1 (newer instruction) (WPM → DQM → OTM → BTM → DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 1665. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A ⁽¹⁾
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A ⁽¹⁾
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM — PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM — PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message ⁽²⁾
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM — Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

74.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

74.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

74.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

74.8.8 Program Trace Sync Message Priority Loss Response

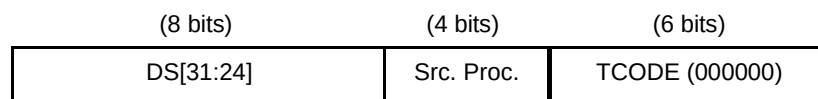
If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

74.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

74.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:



Fixed length = 18 bits

Figure 1793. Debug Status message format

74.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note: The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see [Table 1639](#) and [Table 1640](#)):

(6 bits)	(4 bits)	(4 bits)	(6 bits)
Error Code	Error Type	Src. Proc.	TCODE (001000)

Fixed length = 20 bits

Figure 1794. Error message format

74.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

74.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

74.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z420n3) processors contain a *PowerISA 2.06* defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note: The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

Figure 1795. Nexus 3 Process ID Register (NPIDR)

Nexus 3 Process ID																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR — 517; Read/Write; Reset: 0

Note: OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the DC1_{OTS} control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and DC1_{OTS} indicates PID0 should be used, or when the NPIDR register is written to by the processor and DC1_{OTS} indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting DC1_{POTD}.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15–46 bits

Figure 1796. Ownership Trace Message format

74.12 program Trace

This section details the program trace mechanism supported by Nexus3 for the e200z420n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z420n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

74.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. (**evsel**)Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

74.12.1.1 Indirect Branch Message instructions

[Table 1666](#) shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 1666. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl
System Call / Trap exceptions taken	se_sc, tw
Return from interrupts / exceptions	se_rfi, se_rfc, se_rfdi
Exit from reset with Program Trace Enabled	Indirect branch with Sync, target address is initial instruction, count = 1

74.12.1.2 Direct Branch Message Instructions

[Table 1667](#) shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 1667. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	b, ba, bl, bla, bc, bca, bcl, bcla, se_b, se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync

74.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken. (**evsel**)

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

74.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of “events” that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

74.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages — Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages — Indirect Branch History, and Indirect Branch History with Synchronization Messages.

74.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If DC1_{PTM} is set to ‘1’, indirect branch information is messaged out in the following format:

(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

Figure 1797. Indirect branch message (history) format

74.12.2.2 Indirect branch messages (traditional)

If DC1_{PTM} is cleared to ‘0’, indirect branch information is messaged out in the following format:

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

Figure 1798. Indirect branch message format

74.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

(1–8 bits)	(4 bits)	(6 bits)
Sequence Count	Src. Proc.	TCODE (000011)

Max length = 18 bits; Min length = 11 bits

Figure 1799. Direct branch message format

Note: When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

74.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

74.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

74.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-

shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false. For the **evsel** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 1668. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	bcctr, bcctrl, bclr, bclrl
Not taken direct branches	0	b, ba, bc, bca, bla, bcla, bl, bcl
Taken direct branches	1	b, ba, bc, bca, bla, bcla, bl, bcl⁽¹⁾
evsel instruction	00,01,10, or 11	evsel

1. If the EVCODE for direct branch function calls is not masked in DC4, taken bl and bcl instructions will generate Program Correlation Messages and will not be logged in the history buffer.

74.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

74.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow; or, if program trace is in history mode, since the last instruction that recorded history information.

(1–32 bits)	(4 bits)	(4 bits)	(6 bits)
RDATA	RCODE	Src. Proc.	TCODE (011011)

Max length = 46 bits; Min length = 15 bits

Figure 1800. Resource Full Message format

[Table 1669](#) shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 1669. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch History. This type of packet is terminated by a stop bit set to 1 after the last history bit.

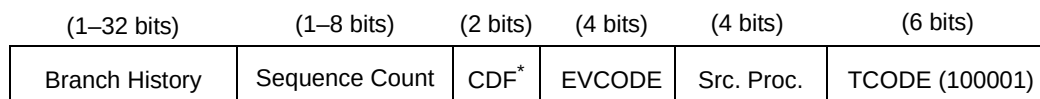
74.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a “Branch and Link” instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to $MSR_{PMM}=0$ and $DC4_{PTMARK}=1$.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to [Table 1642](#) for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:



Max length = 56 bits; Min length = 18 bits

* CDF= 01,

EVCODE = Any but 0101, 1100

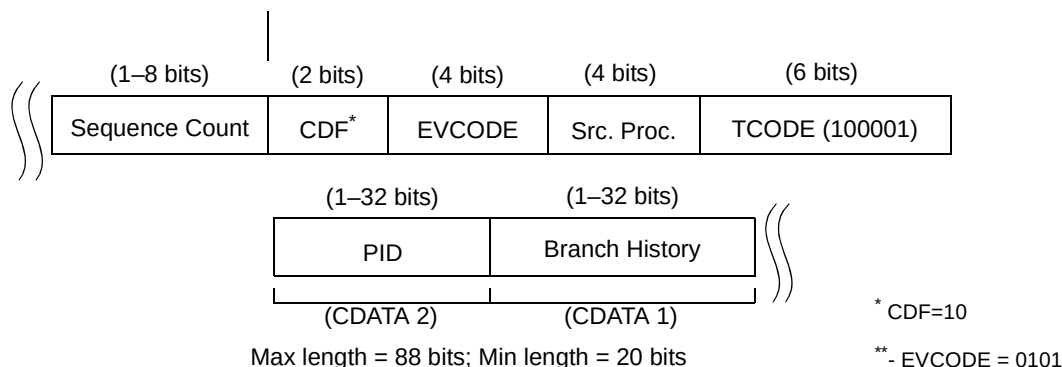


Figure 1801. Program Correlation message formats

74.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

74.12.6 Program Trace Overflow Error messages

an Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

74.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see

[Table 1670](#)):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow '1'$ with $DC4_{PTMARK} = '1'$.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Figure 1802. Program Trace Synchronization message format

Exception conditions that result in Program Trace Synchronization are summarized in [Table 1670](#).

Table 1670. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (j_trst_b), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (p_reset_b), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a nex_evti_b assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first

instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

74.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see [Table 1670](#)):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync messages is as follows:

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

Figure 1803. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. Messages is as follows:

(1–32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Figure 1804. Indirect Branch History w/ Sync message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in [Table 1671](#).

Table 1671. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync. Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the <i>next</i> Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCMPIDMSG → OTM → BTM → DTM) → Instruction 1 (WPM → DQM → OTM → BTM → DTM), where instruction 0 is the oldest instruction. A BTM Message from Instruction 1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

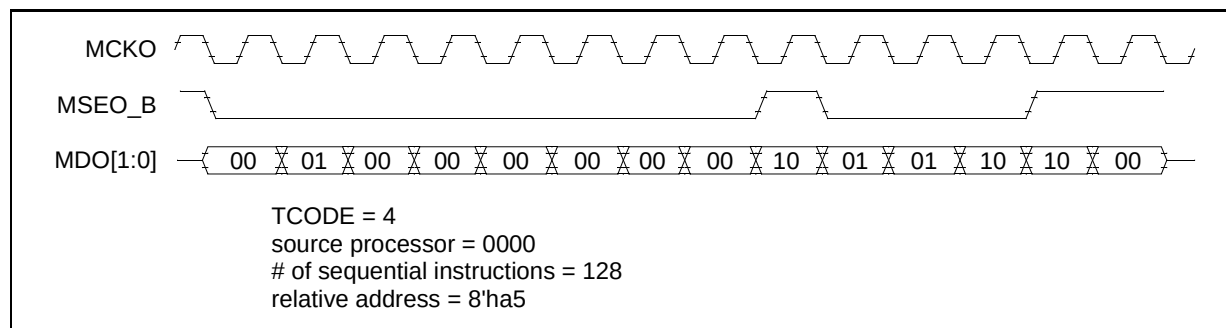
74.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PTMM} bit and the setting of DC4_{PTMARK}

74.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

Figure 1805. Program Trace — Indirect Branch message (Traditional)



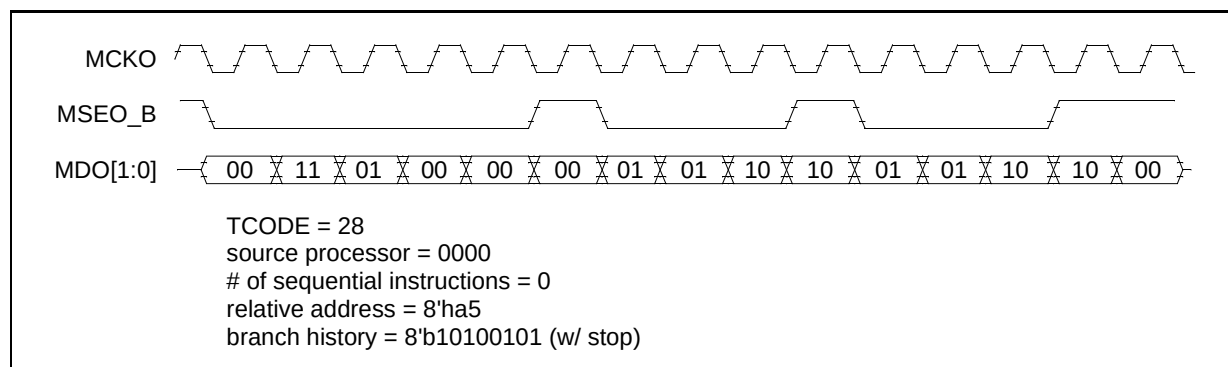


Figure 1806. Program Trace — Indirect Branch message (history)

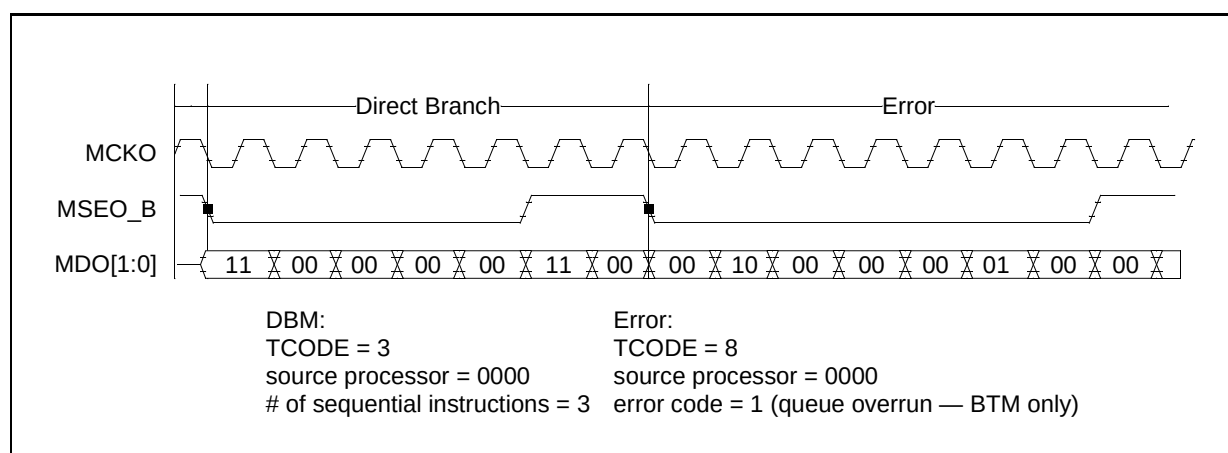


Figure 1807. Program Trace — Direct Branch (traditional) & Error messages

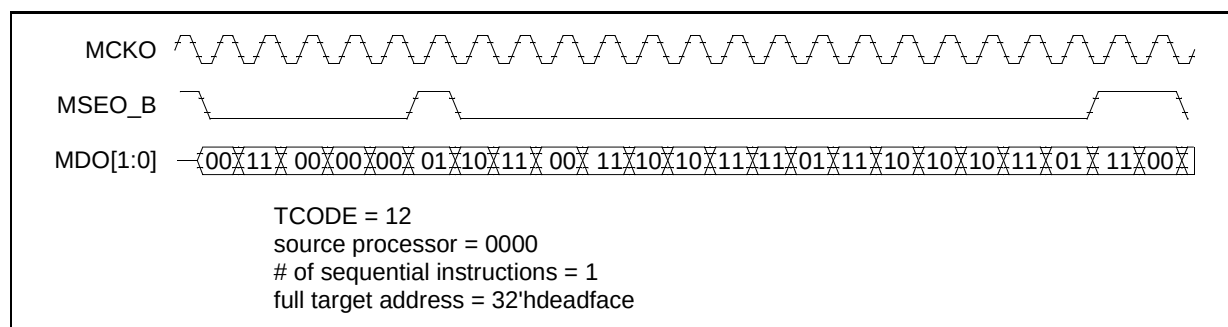


Figure 1808. Program Trace — Indirect Branch w/ Sync message

74.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

74.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note: Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z420n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**)

74.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization, and Error Messages.

74.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace message. Data Write message information is messaged out in the following format:

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 1809. Data Write message format

74.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

Figure 1810. Data Read message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Note: . * Core (e200z420n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Core (e200z420n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

74.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see [Table 1672](#)):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

Figure 1811. Data Write/Read with Sync message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

Exception conditions that result in Data Trace Synchronization are summarized in [Table 1672](#).

Table 1672. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (j_trst_b), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a nex_evti_b assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCMPIDMSG → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

74.13.3 DTM operation

74.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

74.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

74.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and DC4_{DTMARK}.

74.13.3.4 Bus cycle special cases

Table 1673. Bus cycle special cases

Special case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{\text{TEA}}$) ⁽¹⁾	Data Trace Message discarded
Bus cycle completed without error ⁽¹⁾	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC _{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) — both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note: For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z420n3) Core Debug Support chapter.)

Note: A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

74.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

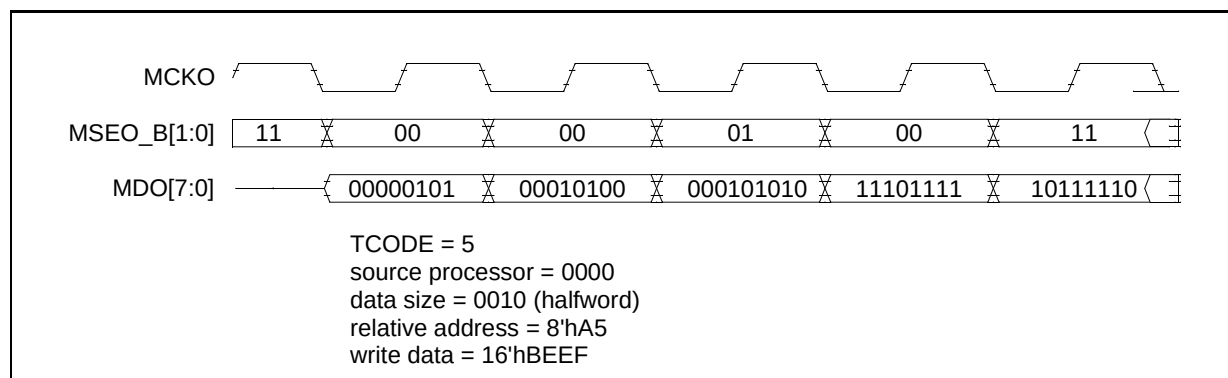


Figure 1812. Data Trace — Data Write message

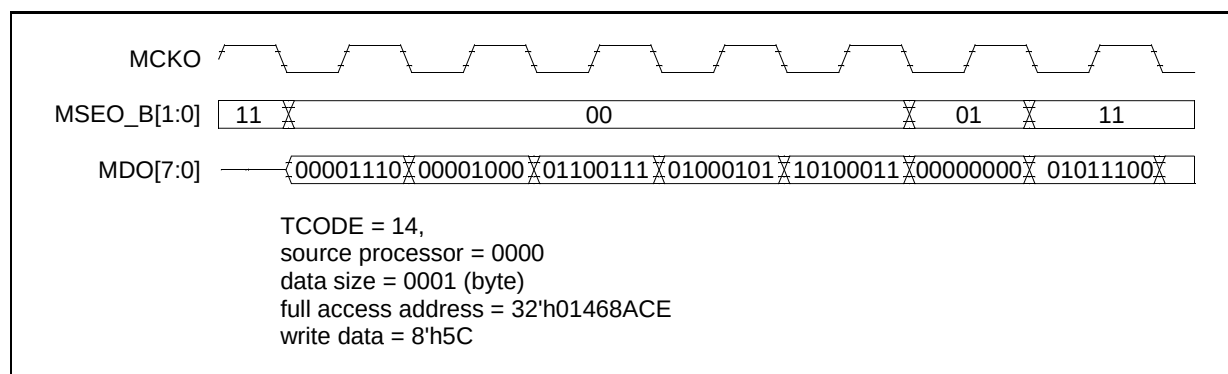


Figure 1813. Data Trace — Data Read w/ Sync message

74.14 Data Acquisition messaging

this section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

74.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

74.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

74.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the “External Debug Resource Allocation Control Register (EDBRAC0)” section in the Core (e200z420n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

(1–32 bits)	(8 bits)	(4 bits)	(6 bits)
DQDATA	DQTAG	Src. Proc.	TCODE (011011)

Max length = 50 bits; Min length = 19 bits

Figure 1814. Data Acquisition message format

74.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z420n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the WMSK_{WEM} field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See [Table 1677](#) for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

Figure 1815. Watchpoint message format

(1–30 bits)	(4 bits)	(6 bits)
Watchpoint Source	Src. Proc.	TCODE (001111)

Variable length = 11–40 bits

The Watchpoint Source message field will contain a ‘1’ for each asserted watchpoint. Leading zeros are truncated.

Table 1674. Watchpoint source encoding

Watchpoint source (1–30 bits)	Watchpoint description
00000000000000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1	Watchpoint #0 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x	Watchpoint #1 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx	Watchpoint #2 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx	Watchpoint #3 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx	Watchpoint #4 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx	Watchpoint #5 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx	Watchpoint #6 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #7 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx	Watchpoint #8 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx	Watchpoint #9 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx	Watchpoint #10 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #11 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #12 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #13 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #14 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #15 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #16 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #17 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #18 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #19 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #20 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #21 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #22 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #23 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #24 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #25 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #26 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #27 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #28 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #29 enabled for WTM



74.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

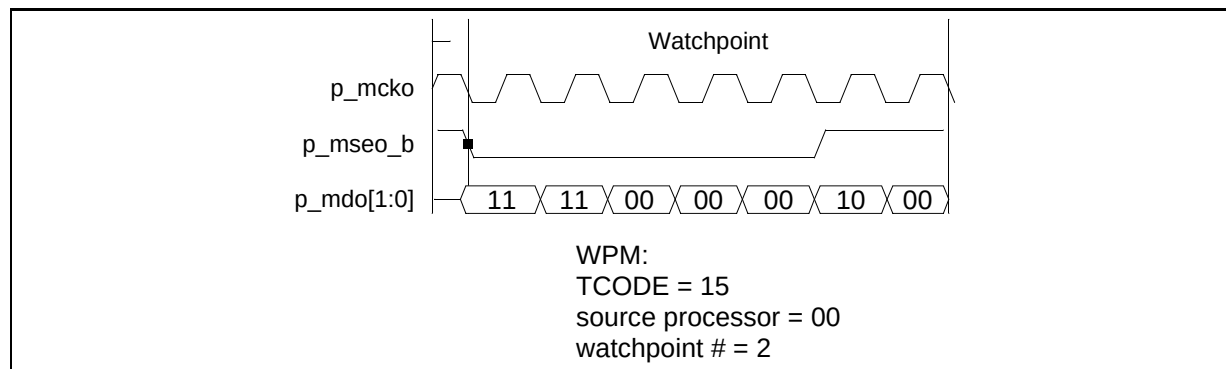


Figure 1816. Watchpoint Message & Watchpoint Error message

74.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a “window” for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect.

[Table 1675](#) details the external hardware trigger signals.

Table 1675. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

Table 1675. External Hardware Trigger Signals (continued)

Name	Description
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0→1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

74.17 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The RWCS_{ATTR} field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note: *Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.*

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

74.17.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure as follows:
 - a) Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure the bits as follows:
 - a) Access Control (AC) → 1b'1 (to indicate start access)
 - b) Map Select (MAP) → 3b'000 (primary memory map)
 - c) Access Priority (PR) → 2b'11 (highest priority)
 - d) Read/Write (RW) → 1b'1 (write access)
 - e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
 - f) Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure as follows:
 - a) Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see [Table 1677](#) for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note: Only the **nex_ahb_start**, **nex_rdy_b**, and **nex_err_b** pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

74.17.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in [Section 74.17.1, Single write access](#) to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has

been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.

3. If AC has not been cleared due to an error, repeat Step 3 in [Section 74.17.1, Single write access](#) until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

74.17.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure as follows:
 - a) Read Address → 32h'xxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure the bits as follows:
 - a) Access Control (AC) → 1b'1 (to indicate start access)
 - b) Map Select (MAP) → 3b'000 (primary memory map)
 - c) Access Priority (PR) → 2b'11 (highest priority)
 - d) Read/Write (RW) → 1b'0 (read access)
 - e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
 - f) Access Count (CNT) → 14h'0000 or 14h'0001 (single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see [Table 1677](#) for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#).

Note: Only the **nex_ahb_start**, **nex_rdy_b**, and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

74.17.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in [Section 74.17.3, Single read access](#) to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and

asserts the **nex_rdy_b** pin (see [Table 1677](#) for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.

3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS Register, and asserts the **nex_rdy_b** pin (see [Table 1677](#) for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#).
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in [Section 74.17.3, Single read access](#) until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note: The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

74.17.5 Error handling

The Nexus 3 module handles various error conditions as follows:

74.17.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared).
2. The ERR bit in the RWCS Register is set.
3. The Error Message is sent (TCODE = 8) indicating Read/Write error.

74.17.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
 - a) If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
 - b) If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

74.17.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

(5 bits)	(4 bits)	(6 bits)
Error Code (00011)	Src. Proc.	TCODE (001000)

Fixed length = 15 bits

Figure 1817. Error message format

74.18 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

74.18.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in [Table 1677](#). The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see [Section 74.5, Nexus 3 Register Access via JTAG/OnCE](#) for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the “JTAG/OnCE Pins” section of the Core (e200z420n3) Core Debug Support chapter.

Table 1676. JTAG pins for Nexus 3

JTAG pins	Input/ Output	Description of JTAG pins (included in Nexus 1)
j_tdo	O	The Test Data Output (j_tdo) pin is the serial output for test instructions and data. j_tdo is three-stateable and is actively driven in the “Shift-IR” and “Shift-DR” controller states. j_tdo changes on the falling edge of j_tclk .
j_tdi	I	The Test Data Input (j_tdi) pin receives serial test instruction and data. TDI is sampled on the rising edge of j_tclk .
j_tms	I	The Test Mode Select (j_tms) input pin is used to sequence the OnCE controller state machine. j_tms is sampled on the rising edge of j_tclk .
j_tclk	I	The Test Clock (j_tclk) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (j_trst_b) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in [Table 1677](#).

Table 1677. Nexus 3 auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (nex_mcko) is a free running output clock to development tools for timing of nex_mdo[n:0] & nex_mseo_b[1:0] pin functions. nex_mcko is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (nex_mdo[n:0]) are output pins used for OTM, BTM, and DTM. External latching of nex_mdo[n:0] shall occur on the rising edge of the Nexus3 clock (nex_mcko).
nex_mseo_b[1:0]	O	Message Start/End Out (nex_mseo_b[1:0]) are output pins that indicate when a message on the nex_mdo[n:0] pins has started, when a variable length packet has ended, and when the message has ended. External latching of nex_mseo_b[1:0] shall occur on the rising edge of the Nexus3 clock (nex_mcko). One or two pin MSEO functionality is determined at integration time per SOC implementation
nex_rdy_b	O	Ready (nex_rdy_b) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, nex_rdy_b remains deasserted
nex_err_b	O	Error (nex_err_b) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, nex_err_b remains deasserted

Table 1677. Nexus 3 auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_evto_b	O	Event Out (nex_evto_b) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. nex_evto_b is held asserted for one (1) cycle of nex_mcko : 1) One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2) Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (nex_evti_b) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1) Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2) Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (nex_wevto[3:0]) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. nex_wevto[3:0] is held asserted for one (1) cycle of nex_mcko .
nex_ext_src_id[0:3]	I	nex_ext_src_id[0:3] is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time
nex_sfwcntl_en]	I	nex_sfwcntl_en is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to [Section 74.20, Auxiliary port arbitration](#), for detail on Nexus port arbitration.

Table 1678. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (nex_aux_req[1:0]) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 1680 .)
nex_aux_busy	O	Nexus Auxiliary Busy (nex_aux_busy) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant (nar_aux_grant) is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (ext_multi_nex_sel) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

74.18.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in [Table 1679](#). Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the SPC57EM80xx.

Table 1679. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

[Figure 1818](#) illustrates the state diagram for single pin MSEO transfers.

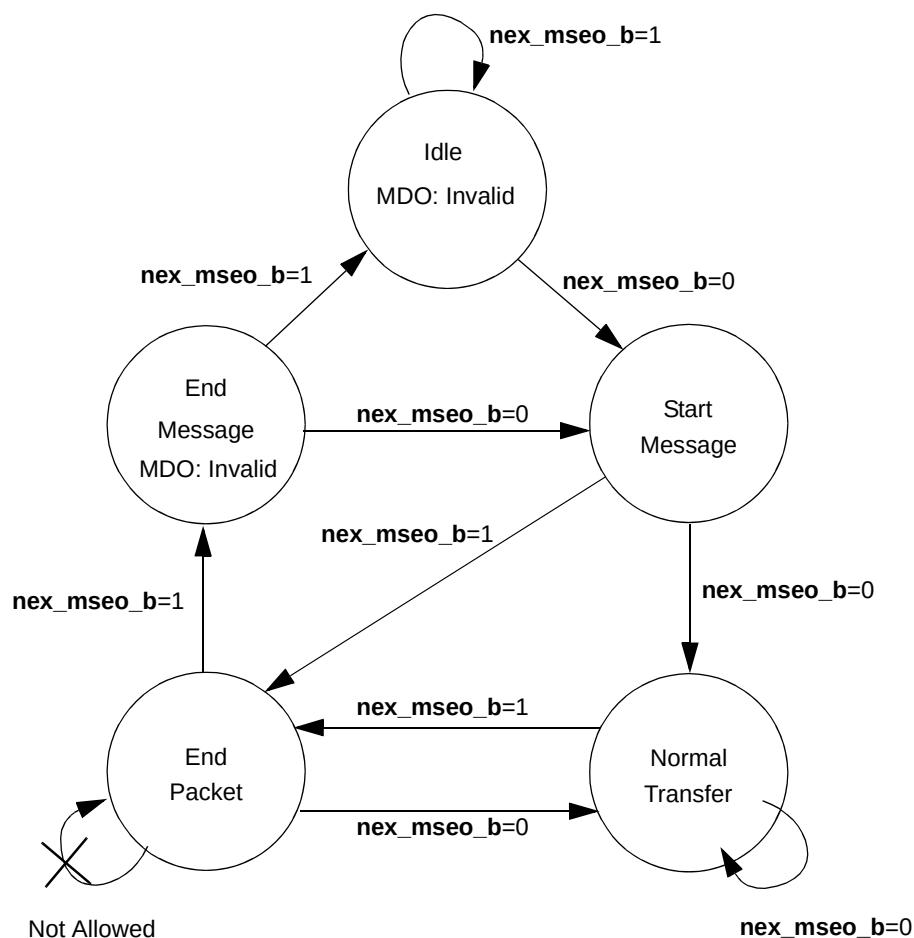


Figure 1818. Single pin MSEO transfers

Note that the “End Message” state does not contain valid data on the `nex_mdo[n:0]` pins. Also, It is not possible to have two consecutive “End Packet” messages. This implies the minimum packet size for a variable length packet is 2x the number of `nex_mdo[n:0]` pins. This ensures that a false end of message state is not entered by emitting two consecutive ‘1’s on the `nex_mseo_b` pin before the actual end of message.

[Figure 1819](#) illustrates the state diagram for dual pin MSEO transfers.

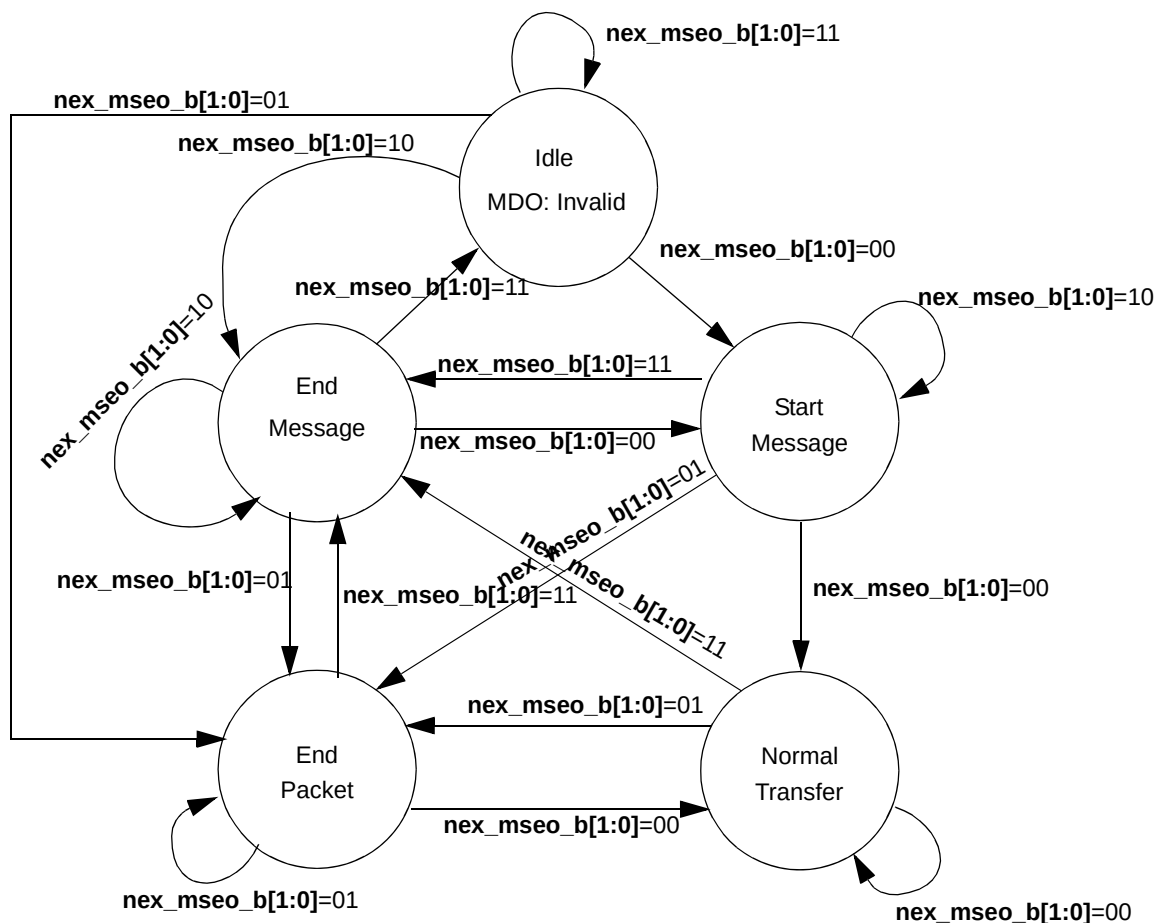


Figure 1819. Dual pin MSEO transfers

The dual pin MSEO option is more robust than the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive “End Packet” states. This can be an advantage when small, variable sized packets are transferred.

Note: The “End Message” state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

74.19 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is

impossible to know which bit is from the last packet and which bit is from the next packet.)

- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

74.20 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level. The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in [Table 1680](#) below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in [Section 74.18.2, Pin protocol](#). The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the MSEO state machine reaches the “End Message” state.

Table 1680. MDO request encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

74.21 Examples

The following are examples of Program Trace and Data Trace Messages.

[Table 1681](#) illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. [Table 1682](#) illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 1681. Indirect branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 1682. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A P	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

[Table 1683](#) and [Table 1684](#) illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 1683. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 1684. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

[Table 1685](#) illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and [Table 1686](#) illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 1685. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 1686. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

74.22 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the SPC57EM80xx Data Sheet.

74.23 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

74.23.1 JTAG sequence for accessing internal Nexus registers

Table 1687. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE → SELECT-DR_SCAN
2	0	SELECT-DR_SCAN → CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR → SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR → EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR → UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR → SELECT-DR_SCAN
8	0	SELECT-DR_SCAN → CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR → SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR → EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR → UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR → RUN-TEST/IDLE (transfer complete — Nexus controller to Reg. Select state)

74.23.2 JTAG sequence for read access of memory-mapped resources

Table 1688. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of nex_rdy_b or nex_err_b pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

74.23.3 JTAG sequence for write access of memory-mapped resources

Table 1689. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value — data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address — data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of nex_rdy_b or nex_err_b pin
8	—	If CNT > 0, go back to Step #5

75 e200z425n3 Nexus 3 Module

The e200z425n3 Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port that is used in conjunction with the JTAG port in these processors.

75.1 Introduction

75.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

75.1.2 Terms and Definitions

[Table 1690](#) contains a set of terms and definitions associated with the Nexus 3 module.

Table 1690. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.

Table 1690. Terms and Definitions (continued)

Term	Description
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	“System-on-a-Chip”. SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase “according to the standard” is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

75.1.3 Feature List

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated.

An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.

- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging
- Auxiliary interface for higher data input/output
 - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
 - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
 - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
 - One (1) Read/Write Error pin (**nex_err_b**) pin
 - One (1) Watchpoint Event output pin (**nex_evto_b**)
 - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
 - One (1) Event In pin (**nex_evti_b**)
 - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)

Note: For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM — maximum number of MDO pins) or Reduced Port Mode (RPM — minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note: The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

75.1.4 Functional Block Diagram

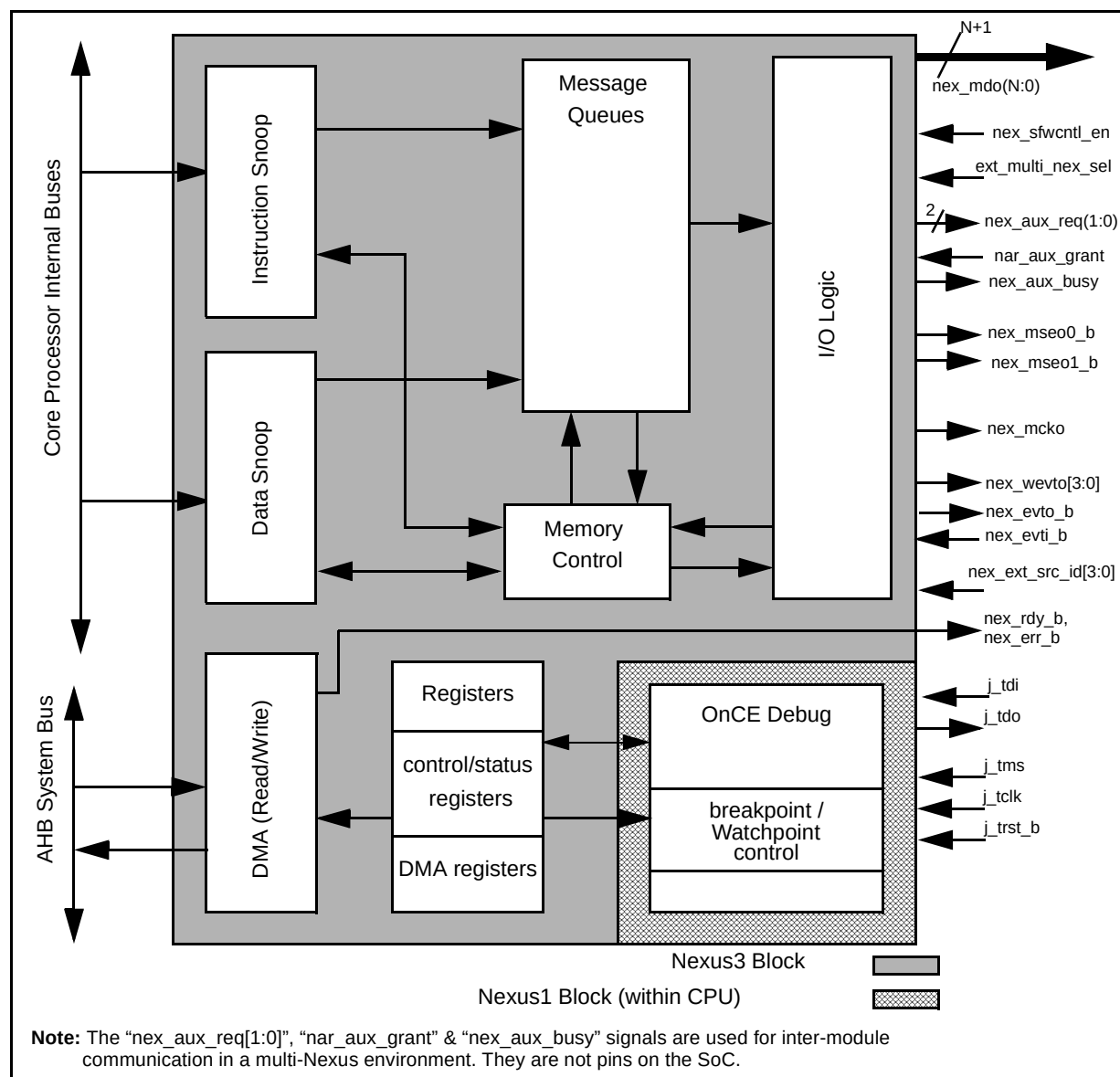


Figure 1820. Nexus 3 functional block diagram

75.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the

Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test-Logic-Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

75.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

75.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in [Table 1691](#).

Table 1691. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace — Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace — Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
Data Trace — Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 1696)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace — Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 1696)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)

Table 1691. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from DEVENT _{DQTAG} register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace — Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace — Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace — Indirect Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)

Table 1691. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Trace — Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 1696)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace — Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 1696)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 1694) — indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section 75.12.4, Resource Full Messages.)
Program Trace — Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	1	32	HIST	variable	branch / predicate instruction history (see Section 75.12.1, Branch Trace Messaging types)

Table 1691. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Program Trace — Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Section 75.12.1, Branch Trace Messaging types.)
Program Trace — Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 1695)
	2	2	CDF	fixed	# fields of information in CDATA. 00 — reserved, 01 — one field (CDATA1) (reserved), 10 — two fields (CDATA1 + CDATA2), 11 — three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 — [branch / predicate instruction history] (See Section 75.12.5, Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2— PID info (See Section 75.12.5, Program Correlation Messages.)

Note: Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in [Section 75.12.1, Branch Trace Messaging types](#). If the Branch History method is selected, the shaded TCODEs above will not be messaged out.

[Table 1692](#) shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 1692. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxxx1xx	Program Trace Message(s) Lost

Table 1692. Error Code (ECODE) Encoding (TCODE = 8) (continued)

Error Code	Description
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

[Table 1693](#) shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 1693. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 – 1111	Reserved

[Table 1694](#) shows the encodings used for resource codes for certain messages.

Table 1694. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

[Table 1695](#) shows the event code encodings used for certain messages.

Table 1695. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010–0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via mtspr PID0/NPIDR

Table 1695. Event Code (EVCODE) Encoding (TCODE = 33) (continued)

Event Code	Description
0110–1000	Reserved for future functionality
1001	Begin masking of program trace messages due to $MSR_{PMM} = 0$ and $DC4_{PTMARK} = 1$
1010	Branch and link occurrence (direct branch function call)
1011–1111	Reserved for future functionality

[Table 1696](#) shows the data trace size encodings used for certain messages.

Table 1696. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 — no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001–1111	Reserved

75.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#) and [Section 75.6, Nexus 3 Register Access via Software](#) for details on Nexus 3 register access.

Note: Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

[Table 1697](#) details the register map for the Nexus 3 module.

Table 1697. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/Write	Read Address	Write Address	DCR # ⁽¹⁾
Client Select Control (CSC) ⁽²⁾	0x1	R	0x02	—	
Port Configuration Register (PCR) ⁽³⁾	PCR_INDEX ²	R/W	—	—	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369

Table 1697. Nexus 3 Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/Write	Read Address	Write Address	DCR # ⁽¹⁾
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	—
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	—
Reserved	0x8	R/W	0x18	0x19	—
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	—
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	—
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	—
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 → 0x2F	—	0x28→0x5E	0x29→0x5F	—
Development Status (DS)	0x30	R	0x60	—	409
Reserved	0x31	R/W	0x62	0x63	—
Overflow Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	—	0x68	0x69	—
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 → 0x3F	—	0x72→0x7E	0x73→0x7F	—

1. Software access via the **mfdcr** and **mtocr** instructions use these values for the DCR number. Software writes to these registers via **mtocr** when **nex_sfvcnt1_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

75.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.

Figure 1821. Client Select Control (CSC) register

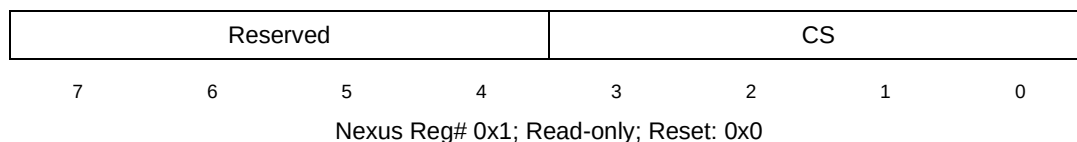


Table 1698. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx Nexus client (SoC level)

75.4.2 Port Configuration Register (PCR) — reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.

Figure 1822. Port Configuration Register (PCR)

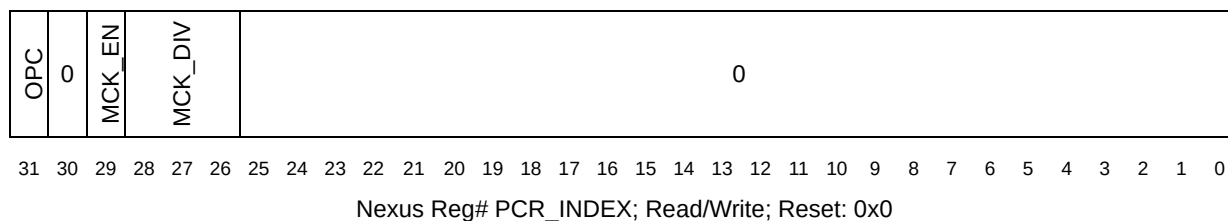


Table 1699. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# nex_mdo[n:0] pins defined by SOC) 1 Full Port Mode configuration (max# nex_mdo[n:0] pins defined by SOC)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 nex_mcko is disabled 1 nex_mcko is enabled

Table 1699. PCR field descriptions (continued)

Bit	Name	Description
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 nex_mcko is 1x processor clock freq. 001 nex_mcko is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 nex_mcko is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 nex_mcko is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note: The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

75.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in [Figure 1823](#) and its fields are described in [Table 1700](#).

Figure 1823. Development Control 1 (DC1) register

OPC	MCK_DIV	0	PTM	0	OTS	0	POTD	TSEN	EOC	EIC	0	TM																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x2; DCR: 368; Read/Write; Reset: 0x0

Table 1700. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 — Reduced Port Mode configuration (min# nex_mdo[n:0] pins defined) 1 — Full Port Mode configuration (max# nex_mdo[n:0] pins defined)
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 — nex_mcko is 1x processor clock freq. 01 — nex_mcko is 1/2x processor clock freq. 10 — nex_mcko is 1/4x processor clock freq. 11 — nex_mcko is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 — Program Trace uses traditional Branch Messages 1 — Program Trace uses Branch History Messages

Table 1700. DC1 field descriptions (continued)

Bits	Name	Description
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 — PID0 data is transmitted within Ownership Trace Messages 1 — Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 — Periodic Ownership Trace message events are enabled 1 — Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable — (not implemented, write to 00) 00 — Timestamp is disabled
11:10	EOC	EVTO Control 00 — nex_evto_b upon occurrence of Watchpoints (configured in DC2 and DC3) 01 — nex_evto_b upon entry into Debug Mode 1x — Reserved
9:8	EIC	EVTI Control 00 — nex_evti_b is used for synchronization (Program Trace Sync msg is generated) 01 — nex_evti_b is used for Debug request 10 — nex_evti_b is disabled 1x — Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode ⁽¹⁾ 000000 — All Trace Disabled xxxxx1 — Ownership Trace enabled xxxx1x — Data Trace enabled xxx1xx — Program Trace enabled xx1xxx — Watchpoint Trace enabled x1xxxx — Reserved 1xxxxx — Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to [Section 75.4.7, Watchpoint Trigger \(WT, PTSTC, PTETC, DTSTC, DTETC\) registers](#) for more information on watchpoint triggering. Refer to [Section 75.16, External Hardware Trigger Controls](#) for more information on external hardware triggering.

Note: The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) **MUST ONLY** be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

75.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z425n3) Core Debug Support chapter.

Development Control Register 2 is shown in [Figure 1824](#) and its fields are described in [Table 1701](#).

Figure 1824. Development Control 2 (DC2) register

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	EWC
-----------	-----------	-----------	-----------	-----

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x3; DCR: 369; Read/Write; Reset: 0x0

Table 1701. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 — No Watchpoints #0–14 trigger nex_wevto[3] 0001 — Watchpoint #0 triggers nex_wevto[3] 0010 — Watchpoint #1 triggers nex_wevto[3] 0011 — Watchpoint #2 triggers nex_wevto[3] 0100 — Watchpoint #3 triggers nex_wevto[3] 0101 — Watchpoint #4 triggers nex_wevto[3] 0110 — Watchpoint #5 triggers nex_wevto[3] 0111 — Watchpoint #6 triggers nex_wevto[3] 1000 — Watchpoint #7 triggers nex_wevto[3] 1001 — Watchpoint #8 triggers nex_wevto[3] 1010 — Watchpoint #9 triggers nex_wevto[3] 1011 — Watchpoint #10 triggers nex_wevto[3] 1100 — Watchpoint #11 triggers nex_wevto[3] 1101 — Watchpoint #12 triggers nex_wevto[3] 1110 — Watchpoint #13 triggers nex_wevto[3] 1111 — Watchpoint #14 triggers nex_wevto[3]
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 — No Watchpoints #0–14 trigger nex_wevto[2] 0001 — Watchpoint #0 triggers nex_wevto[2] 0010 — Watchpoint #1 triggers nex_wevto[2] 0011 — Watchpoint #2 triggers nex_wevto[2] 0100 — Watchpoint #3 triggers nex_wevto[2] 0101 — Watchpoint #4 triggers nex_wevto[2] 0110 — Watchpoint #5 triggers nex_wevto[2] 0111 — Watchpoint #6 triggers nex_wevto[2] 1000 — Watchpoint #7 triggers nex_wevto[2] 1001 — Watchpoint #8 triggers nex_wevto[2] 1010 — Watchpoint #9 triggers nex_wevto[2] 1011 — Watchpoint #10 triggers nex_wevto[2] 1100 — Watchpoint #11 triggers nex_wevto[2] 1101 — Watchpoint #12 triggers nex_wevto[2] 1110 — Watchpoint #13 triggers nex_wevto[2] 1111 — Watchpoint #14 triggers nex_wevto[2]

Table 1701. DC2 field descriptions (continued)

Bits	Name	Description
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 — No Watchpoints #0–14 trigger nex_wevto[1] 0001 — Watchpoint #0 triggers nex_wevto[1] 0010 — Watchpoint #1 triggers nex_wevto[1] 0011 — Watchpoint #2 triggers nex_wevto[1] 0100 — Watchpoint #3 triggers nex_wevto[1] 0101 — Watchpoint #4 triggers nex_wevto[1] 0110 — Watchpoint #5 triggers nex_wevto[1] 0111 — Watchpoint #6 triggers nex_wevto[1] 1000 — Watchpoint #7 triggers nex_wevto[1] 1001 — Watchpoint #8 triggers nex_wevto[1] 1010 — Watchpoint #9 triggers nex_wevto[1] 1011 — Watchpoint #10 triggers nex_wevto[1] 1100 — Watchpoint #11 triggers nex_wevto[1] 1101 — Watchpoint #12 triggers nex_wevto[1] 1110 — Watchpoint #13 triggers nex_wevto[1] 1111 — Watchpoint #14 triggers nex_wevto[1]

Table 1701. DC2 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 — No Watchpoints #0–14 trigger nex_wevto[0] 0001 — Watchpoint #0 triggers nex_wevto[0] 0010 — Watchpoint #1 triggers nex_wevto[0] 0011 — Watchpoint #2 triggers nex_wevto[0] 0100 — Watchpoint #3 triggers nex_wevto[0] 0101 — Watchpoint #4 triggers nex_wevto[0] 0110 — Watchpoint #5 triggers nex_wevto[0] 0111 — Watchpoint #6 triggers nex_wevto[0] 1000 — Watchpoint #7 triggers nex_wevto[0] 1001 — Watchpoint #8 triggers nex_wevto[0] 1010 — Watchpoint #9 triggers nex_wevto[0] 1011 — Watchpoint #10 triggers nex_wevto[0] 1100 — Watchpoint #11 triggers nex_wevto[0] 1101 — Watchpoint #12 triggers nex_wevto[0] 1110 — Watchpoint #13 triggers nex_wevto[0] 1111 — Watchpoint #14 triggers nex_wevto[0]
15:0	EWC	EVTO Watchpoint Configuration ⁽¹⁾ 0000000000000000 No Watchpoints #0–15 trigger nex_evto_b xxxxxxxxxxxxxxxx1 Watchpoint #0 triggers nex_evto_b xxxxxxxxxxxxxxxx1x Watchpoint #1 triggers nex_evto_b xxxxxxxxxxxxxxxx1xx Watchpoint #2 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxx Watchpoint #3 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxxx Watchpoint #4 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxxxx Watchpoint #5 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxxxxx Watchpoint #6 triggers nex_evto_b xxxxxxxxxxxxxxxx1xxxxxxx Watchpoint #7 triggers nex_evto_b xxxxxxxx1xxxxxxx Watchpoint #8 triggers nex_evto_b xxxxxx1xxxxxxx Watchpoint #9 triggers nex_evto_b xxxxx1xxxxxxx Watchpoint #10 triggers nex_evto_b xxxx1xxxxxxx Watchpoint #11 triggers nex_evto_b xxx1xxxxxxx Watchpoint #12 triggers nex_evto_b xx1xxxxxxx Watchpoint #13 triggers nex_evto_b x1xxxxxxx Watchpoint #14 triggers nex_evto_b 1xxxxxxx Watchpoint #15 triggers nex_evto_b

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in [Figure 1825](#) and its fields are described in [Table 1702](#).

Figure 1825. Development Control 3 (DC3) register

WEVTO[3]C																																WEVTO[2]C								WEVTO[1]C								WEVTO[0]C								0				EWC															
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																																																																											
Nexus Reg# 0x4; DCR: 370; Read/Write; Reset: 0x0																																																																											

Table 1702. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #15–#26 trigger nex_wevto[3] 0001 Watchpoint #15 triggers nex_wevto[3] 0010 Watchpoint #16 triggers nex_wevto[3] 0011 Watchpoint #17 triggers nex_wevto[3] 0100 Watchpoint #18 triggers nex_wevto[3] 0101 Watchpoint #19 triggers nex_wevto[3] 0110 Watchpoint #20 triggers nex_wevto[3] 0111 Watchpoint #21 triggers nex_wevto[3] 1000 Watchpoint #22 triggers nex_wevto[3] 1001 Watchpoint #23 triggers nex_wevto[3] 1010 Watchpoint #24 triggers nex_wevto[3] 1011 Watchpoint #25 triggers nex_wevto[3] 1100 Watchpoint #26 triggers nex_wevto[3] 11x1 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #15–#26 trigger nex_wevto[2] 0001 Watchpoint #15 triggers nex_wevto[2] 0010 Watchpoint #16 triggers nex_wevto[2] 0011 Watchpoint #17 triggers nex_wevto[2] 0100 Watchpoint #18 triggers nex_wevto[2] 0101 Watchpoint #19 triggers nex_wevto[2] 0110 Watchpoint #20 triggers nex_wevto[2] 0111 Watchpoint #21 triggers nex_wevto[2] 1000 Watchpoint #22 triggers nex_wevto[2] 1001 Watchpoint #23 triggers nex_wevto[2] 1010 Watchpoint #24 triggers nex_wevto[2] 1011 Watchpoint #25 triggers nex_wevto[2] 1100 Watchpoint #26 triggers nex_wevto[2] 11x1 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #15–#29 trigger nex_wevto[1] 0001 Watchpoint #15 triggers nex_wevto[1] 0010 Watchpoint #16 triggers nex_wevto[1] 0011 Watchpoint #17 triggers nex_wevto[1] 0100 Watchpoint #18 triggers nex_wevto[1] 0101 Watchpoint #19 triggers nex_wevto[1] 0110 Watchpoint #20 triggers nex_wevto[1] 0111 Watchpoint #21 triggers nex_wevto[1] 1000 Watchpoint #22 triggers nex_wevto[1] 1001 Watchpoint #23 triggers nex_wevto[1] 1010 Watchpoint #24 triggers nex_wevto[1] 1011 Watchpoint #25 triggers nex_wevto[1] 1100 Watchpoint #26 triggers nex_wevto[1] 1101 Watchpoint #27 triggers nex_wevto[1] 1110 Watchpoint #28 triggers nex_wevto[1] 1111 Watchpoint #29 triggers nex_wevto[1]

Table 1702. DC3 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 No Watchpoints #15–#29 trigger nex_wevto[0] 0001 Watchpoint #15 triggers nex_wevto[0] 0010 Watchpoint #16 triggers nex_wevto[0] 0011 Watchpoint #17 triggers nex_wevto[0] 0100 Watchpoint #18 triggers nex_wevto[0] 0101 Watchpoint #19 triggers nex_wevto[0] 0110 Watchpoint #20 triggers nex_wevto[0] 0111 Watchpoint #21 triggers nex_wevto[0] 1000 Watchpoint #22 triggers nex_wevto[0] 1001 Watchpoint #23 triggers nex_wevto[0] 1010 Watchpoint #24 triggers nex_wevto[0] 1011 Watchpoint #25 triggers nex_wevto[0] 1100 Watchpoint #26 triggers nex_wevto[0] 1101 Watchpoint #27 triggers nex_wevto[0] 1110 Watchpoint #28 triggers nex_wevto[0] 1111 Watchpoint #29 triggers nex_wevto[0]
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration ⁽¹⁾ 0000000000000000 No Watchpoints #16–#29 trigger nex_evto_b xxxxxxxxxxxxxx1 Watchpoint #16 triggers nex_evto_b xxxxxxxxxxxxxx1x Watchpoint #17 triggers nex_evto_b xxxxxxxxxxxxxx1xx Watchpoint #18 triggers nex_evto_b xxxxxxxxxxxxxx1xxx Watchpoint #19 triggers nex_evto_b xxxxxxxxxxxxxx1xxxx Watchpoint #20 triggers nex_evto_b xxxxxxxxxxxxxx1xxxxx Watchpoint #21 triggers nex_evto_b xxxxxxxxxxxxxx1xxxxxx Watchpoint #22 triggers nex_evto_b xxxxxx1xxxxxxx Watchpoint #23 triggers nex_evto_b xxxxx1xxxxxxx Watchpoint #24 triggers nex_evto_b xxx1xxxxxxx Watchpoint #25 triggers nex_evto_b xxx1xxxxxxx Watchpoint #26 triggers nex_evto_b xx1xxxxxxx Watchpoint #27 triggers nex_evto_b x1xxxxxxx Watchpoint #28 triggers nex_evto_b 1xxxxxxx Watchpoint #29 triggers nex_evto_b

1. EOC bits in DC1 must be programmed to trigger EVTO on Watchpoint occurrence for EWC bits to have any effect.

75.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in [Figure 1826](#) and its fields are described in [Table 1703](#).

Figure 1826. Development Control 4 (DC4) register

PTMARK	DTMARK	0														EVCDM															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x5; DCR: 371; Read/Write; Reset: 0x0

Table 1703. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR _{PMM} for masking program trace messages 1 Mask program trace messages when MSR _{PMM} =‘0’, unmask program trace messages when MSR _{PMM} =‘1’
30	DTMARK	Data Trace Mark 0 Ignore MSR _{PMM} for masking data trace messages 1 Mask data trace messages when MSR _{PMM} =‘0’, unmask data trace messages when MSR _{PMM} =‘1’
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask ⁽¹⁾ 0000000000000000 No EVCODEs masked for Program Correlation Messages xxxxxxxxxxxxxxx1x EVCODE #0 is masked for Program Correlation Messages xxxxxxxxxxxxxxx1x EVCODE #1 is masked for Program Correlation Messages xxxxxxxxxxxxxxx1xx EVCODE #2 is masked for Program Correlation Messages xxxxxxxxxxxxxxx1xxx EVCODE #3 is masked for Program Correlation Messages xxxxxxxxxxxx1xxxx EVCODE #4 is masked for Program Correlation Messages xxxxxxxxxxxx1xxxxx EVCODE #5 is masked for Program Correlation Messages xxxxxxxxxxxx1xxxxxx EVCODE #6 is masked for Program Correlation Messages xxxxxxxxxxx1xxxxxxx EVCODE #7 is masked for Program Correlation Messages xxxxxxxx1xxxxxxx EVCODE #8 is masked for Program Correlation Messages xxxxxx1xxxxxxx EVCODE #9 is masked for Program Correlation Messages xxxxx1xxxxxxx EVCODE #10 is masked for Program Correlation Messages xxx1xxxxxxx EVCODE #11 is masked for Program Correlation Messages xxx1xxxxxxx EVCODE #12 is masked for Program Correlation Messages xx1xxxxxxx EVCODE #13 is masked for Program Correlation Messages x1xxxxxxx EVCODE #14 is masked for Program Correlation Messages 1xxxxxxx EVCODE #15 is masked for Program Correlation Messages

1. Refer to [Table 1695](#) for implemented EVCODEs.

75.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.

Figure 1827. Development Status (DS) register

DBG	LPS	LPC	0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			

Nexus Reg# 0x4; DCR: 409; Read-only; Reset: 0x0

Table 1704. DS field descriptions

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (jd_debug_b signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting State (p_waiting signal asserted) 010 Halted State (p_halted signal asserted) 011 Reserved 1xx Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (p_halted signal asserted) 10 CPU in Stopped state (p_stopped signal asserted) 11 CPU in Waiting state (p_waiting signal asserted)
25:0	—	Reserved for future functionality (read as 0)

75.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related “window” for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For

example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z425n3) Core Debug Support chapter.

Figure 1828. Watchpoint Trigger (WT) register

PTS		PTE		DTS		DTE		0																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0xB; DCR: 375; Read/Write; Reset: 0x0

[Table 1705](#) details the Watchpoint Trigger register fields.

Table 1705. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

Figure 1829. Program Trace Start Trigger Control (PTSTC) register

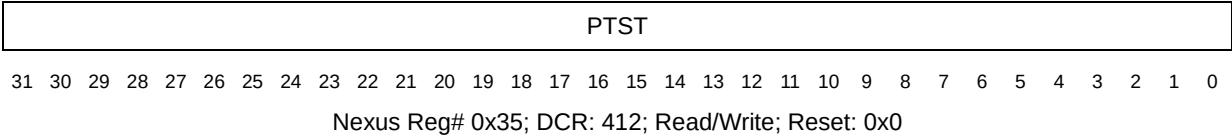


Table 1706 details the PTSTC register fields.

Table 1706. PTSTC field descriptions

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control
		00000000000000000000000000000000 Trigger disabled
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #8
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #9
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #10
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #11
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #12
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #13
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #14
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #15
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #16
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #17
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #18
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #19
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #20
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #21
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #22
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #23
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #24
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #25
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #26
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #27
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #28
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #29
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #30
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

Figure 1830. Program Trace End Trigger Control (PTETC) register

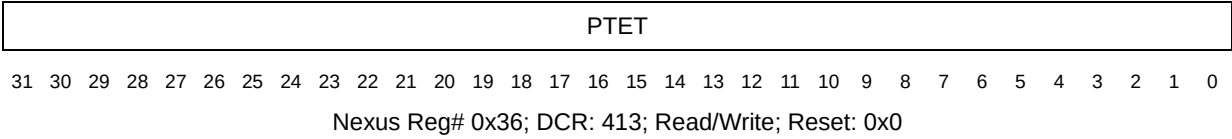


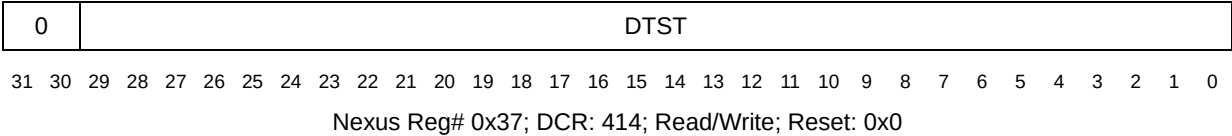
Table 1707 details the PTETC register fields.

Table 1707. PTETC field descriptions

Bits	Name	Description
31:0	PTET	Program Trace End Trigger Control
		00000000000000000000000000000000 Trigger disabled
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #8
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #9
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #10
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #11
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #12
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #13
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #14
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #15
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #16
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #17
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #18
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #19
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #20
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #21
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #22
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #23
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #24
		xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #25
		xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #26
		xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #27
		xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #28
		xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #29
		x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #30
		1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

Figure 1831. Data Trace Start Trigger Control (DTSTC) register



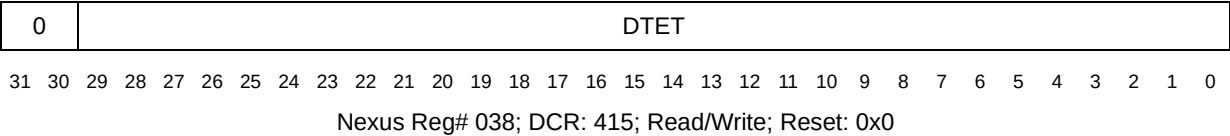
The following table details the DTSTC register fields.

Table 1708. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #16 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #17 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #18 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #19 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #20 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #21 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #22 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #23 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #24 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #25 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #26 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #27 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #28 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

Figure 1832. Data Trace End Trigger Control (DTETC) register



The following table details the DTETC register fields.

Table 1709. DTETC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #16 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #17 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #18 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #19 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #20 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #21 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #22 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #23 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #24 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #25 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #26 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #27 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #28 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #29

75.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages (DC1_{TM} must also be programmed to generate Watchpoint Trace Messages).

Figure 1833. Watchpoint Mask (WMSK) register

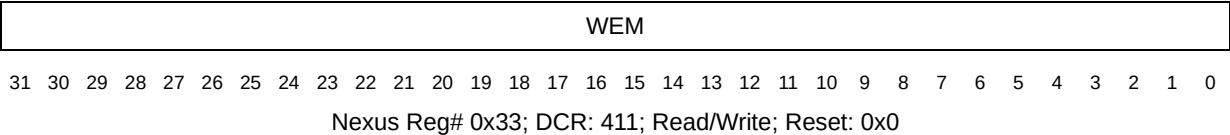


Table 1710 details the Watchpoint Trigger register fields.

Table 1710. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging
		xx1 Watchpoint #0 enabled for WTM
		xx1x Watchpoint #1 enabled for WTM
		xx1xx Watchpoint #2 enabled for WTM
		xx1xxx Watchpoint #3 enabled for WTM
		xx1xxxx Watchpoint #4 enabled for WTM
		xx1xxxxx Watchpoint #5 enabled for WTM
		xx1xxxxxx Watchpoint #6 enabled for WTM
		xx1xxxxxxx Watchpoint #7 enabled for WTM
		xx1xxxxxxxx Watchpoint #8 enabled for WTM
		xx1xxxxxxxxx Watchpoint #9 enabled for WTM
		xx1xxxxxxxxxx Watchpoint #10 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #11 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #12 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #13 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #14 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #15 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #16 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #17 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #18 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #19 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #20 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #21 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #22 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #23 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #24 enabled for WTM
		xx1xxxxxxxxxxx Watchpoint #25 enabled for WTM
		xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #26 enabled for WTM
		xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #27 enabled for WTM
		xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #28 enabled for WTM
		xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #29 enabled for WTM
		x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #30 enabled for WTM
		1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #31 enabled for WTM

75.4.9 Nexus Overrun Control (OVCR) register

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

Figure 1834. Nexus Overrun Control (OVCR) register

0		SPTHOLD		0				SPEN				0		STTHOLD		0								STEN							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x32; DCR: 410; Read/Write; Reset: 0x0

Figure 1835. OVCR field descriptions

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 Suppression threshold is when message queues are 1/4 full 01 Suppression threshold is when message queues are 1/2 full 10 Suppression threshold is when message queues are 3/4 full 11 Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 Suppression is disabled xxxxx1 Ownership Trace message suppression is enabled xxx1xx Data Trace message suppression is enabled xx1xxx Program Trace message suppression is enabled x1xxxx Watchpoint Trace message suppression is enabled 1xxxxx Reserved 1xxxxx Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 Stall threshold is when message queues are 1/4 full 01 Stall threshold is when message queues are 1/2 full 10 Stall threshold is when message queues are 3/4 full 11 Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 Stalling is disabled 1 Stalling is enabled

75.4.10 Data Trace Control (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

Figure 1836. Data Trace Control (DTC) register

RWT1		RWT2		RWT3		RWT4		0								STDC1	STDC2	STDC3	STDC4	0				RC1	RC2	RC3	RC4	DI	0		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0xD; DCR: 376; Read/Write; Reset: 0x0

The following table details the Data Trace Control register fields.

Table 1711. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs

Table 1711. DTC field descriptions (continued)

Bits	Name	Description
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses <i>Note: The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module.</i> <i>Note: The setting of the DI bit also affects the information provided on the data trace port outputs (See the “Data Trace Port Signals” section in the Core (e200z425n3) Core Debug Support chapter.)</i>
2:0	—	Reserved for future functionality (read as 0)

75.4.11 Data Trace Start Address Registers (DTSA1–4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Figure 1837. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0xE; DCR: 377; Read/Write; Reset: 0x0																															

Figure 1838. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0xF; DCR: 378; Read/Write; Reset: 0x0																															

Figure 1839. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0x10; DCR: 379; Read/Write; Reset: 0x0																															

Figure 1840. Data Trace Start Address 4 (DTSA4) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x11; DCR: 380; Read/Write; Reset: 0x0

75.4.12 Data Trace End Address 1–4 (DTEA1–4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Figure 1841. Data Trace End Address 1 register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0x12; DCR: 381; Read/Write; Reset: 0x0																															

Figure 1842. Data Trace End Address 2 register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0x13; DCR: 382; Read/Write; Reset: 0x0																															

Figure 1843. Data Trace End Address 3 register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x14; DCR: 383; Read/Write; Reset: 0x0

Figure 1844. Data Trace End Address 4 register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x15; DCR: 408; Read/Write; Reset: 0x0

[Table 1712](#) illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 1712. Data trace — address range options

Programmed values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA → ← DTEA
DTSA < DTEA	1	← DTSA DTEA →
DTSA > DTEA	N/A	Invalid Range — no trace
DTSA = DTEA	N/A	Invalid Range — no trace

Note: DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are inclusive of the DTSA and DTEA addresses for Range Control settings indicating “within range,” and are exclusive of the DTSA and DTEA addresses for Range Control settings indicating “outside of range.”

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when $DTC_{DI} = 1$, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

75.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes. Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values. The RWCS Register also provides Read/Write Access Status information per [Table 1714](#).

Figure 1845. Read/Write Access Control/Status (RWCS) register

AC	RW	SZ		MAP		PR		ATTR		0		CNT																ERR	DV		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x7; Read/Write⁽¹⁾; Reset: 0x0

1. ERR and DV are read-only

Table 1713. RWCS field descriptions

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access

Table 1713. RWCS field descriptions (continued)

Bits	Name	Description
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 1xx Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map All other values are reserved
RWCS[23:22]	PR ⁽¹⁾	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx p_d_hprot[4] driven to 0 for accesses x1xx p_d_hprot[4] driven to 1 for accesses xx0x p_d_hprot[3] driven to 0 for accesses xx1x p_d_hprot[3] driven to 1 for accesses xxx0 p_d_hprot[2] driven to 0 for accesses xxx1 p_d_hprot[2] driven to 1 for accesses
RWCS[17:16]	—	Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR ⁽²⁾	Read/Write Access Error (see Table 1714)
RWCS[0]	DV ⁽²⁾	Read/Write Access Data Valid (see Table 1714)

1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 1714. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

75.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Figure 1846. Read/Write Access Data (RWD) register

Read/Write Data																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nexus Reg# 0xA; Read/Write; Reset: 0x0																															

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. [Table 1715](#) shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 1715. RWD data placement for transfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	—	—	—	X
Half	x x 0	0 0 1	—	—	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword						
first RWD pass (low order data)	0 0 0	0 1 1	X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

“X” indicates byte lanes with valid data

“—” indicates byte lanes that will contain unused data.

[Table 1716](#) shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 1716. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	AHB[7:0]
Byte @001	0 0 1	—	—	—	AHB[15:8]
Byte @010	0 1 0	—	—	—	AHB[23:16]
Byte @011	0 1 1	—	—	—	AHB[31:24]
Byte @100	1 0 0	—	—	—	AHB[39:32]
Byte @101	1 0 1	—	—	—	AHB[47:40]
Byte @110	1 1 0	—	—	—	AHB[55:48]
Byte @111	1 1 1	—	—	—	AHB[63:56]
Half @000	0 0 0	—	—	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	—	—	AHB[31:24]	AHB[23:16]

Table 1716. RWD byte lane mapping (continued)

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Half @100	1 0 0	—	—	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	—	—	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
first RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
second RWD pass					

Table Notes:
 “—” indicates byte lanes that will contain unused data.

75.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Figure 1847. Read/Write Access Address (RWA) register

Read/Write Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x9; Read/Write; Reset: 0x0

75.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction (“NEXUS3-ACCESS”) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in [Table 1697](#).

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See [Section 75.23, IEEE 1149.1 \(JTAG\) RD/WR sequences](#).)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see [Table 1697](#)), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

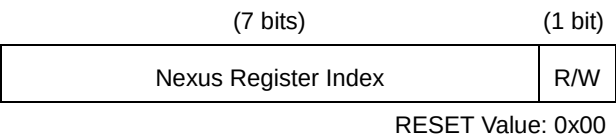


Figure 1848. JTAG DR format

Table 1717. JTAG DR field descriptions

Nexus Register Index	Selected from values in Table 1697
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
- a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the “Capture-DR” state.

b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the “Update-DR” state.

75.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. [Table 1697](#) shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

75.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

75.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.



75.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

75.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) = 0x0003FC01, New Address (A2) = 0x0003F365

Message Generation: A1 = 0000 0000 0000 0011 1111 1100 0000 0001 A2 = 0000 0000 0000 0011 1111 0011 0110 0101 $A1 \oplus A2 = 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 0110\ 0100$ Address Message (M1) = 1111 0110 0100 Address Re-creation: $A1 \oplus M1 = A2$ A1 = 0000 0000 0000 0011 1111 1100 0000 0001 M1 = 0000 0000 0000 0000 0000 1111 0110 0100 A2 = 0000 0000 0000 0011 1111 0011 0110 0101

Figure 1849. Relative address generation and recreation

75.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

75.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

75.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

75.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to [Section 75.4.9, Nexus Overrun Control \(OVCR\) register](#), for complete programming options.

75.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in [Section 75.8.1, Message Queue Overrun](#). Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

75.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

[Table 1718](#) lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in [Table 1718](#).

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction 1 (newer instruction) (WPM → DQM → OTM → BTM → DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 1718. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A ⁽¹⁾
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A ⁽¹⁾
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM — PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM — PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message ⁽²⁾
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM — Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

75.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

75.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

75.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

75.8.8 Program Trace Sync Message Priority Loss Response

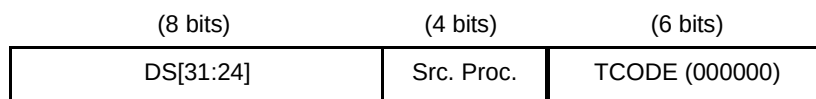
If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

75.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

75.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:



Fixed length = 18 bits

Figure 1850. Debug Status message format

75.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note: The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see [Table 1692](#) and [Table 1693](#)):

(6 bits)	(4 bits)	(4 bits)	(6 bits)
Error Code	Error Type	Src. Proc.	TCODE (001000)

Fixed length = 20 bits

Figure 1851. Error message format

75.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

75.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

75.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z425n3) processors contain a *PowerISA 2.06* defined “Process ID” register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note: The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

Figure 1852. Nexus 3 Process ID Register (NPIDR)

Nexus 3 Process ID																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR — 517; Read/Write; Reset: 0

Note: OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the DC1_{OTS} control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and DC1_{OTS} indicates PID0 should be used, or when the NPIDR register is written to by the processor and DC1_{OTS} indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting DC1_{POTD}.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15–46 bits

Figure 1853. Ownership Trace Message format

75.12 program Trace

This section details the program trace mechanism supported by Nexus3 for the e200z425n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z425n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

75.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Certain instructions (**zvselh**) (**evsel**) generate a pair of predicate bits that are both reported as consecutive bits in the history field. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

75.12.1.1 Indirect Branch Message instructions

[Table 1719](#) shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 1719. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	bcctr , bcctrl , bclr , bclrl , se_bctr , se_bctrl , se_blr , se_brlr
System Call / Trap exceptions taken	se_sc , tw
Return from interrupts / exceptions	se_rfi , se_rfc , se_rfdi
Exit from reset with Program Trace Enabled	Indirect branch with Sync, target address is initial instruction, count = 1

75.12.1.2 Direct Branch Message Instructions

[Table 1720](#) shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 1720. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	b, ba, bl, bla, bc, bca, bcl, bcla, se_b, se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync

75.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken. Certain instructions (**zvselh**) (**evsel**) generate a pair of predicate bits that are both reported as consecutive bits in the history field.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

75.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of “events” that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

75.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages — Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages — Indirect Branch History, and Indirect Branch History with Synchronization Messages.

75.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If DC1_{PTM} is set to ‘1’, indirect branch information is messaged out in the following format:

(1–32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

Figure 1854. Indirect branch message (history) format

75.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

Figure 1855. Indirect branch message format

75.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

(1–8 bits)	(4 bits)	(6 bits)
Sequence Count	Src. Proc.	TCODE (000011)

Max length = 18 bits; Min length = 11 bits

Figure 1856. Direct branch message format

Note: When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

75.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

75.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

75.12.3.2 Branch/Predicate Instruction History (HIST)

If DC1_{PTM} is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false. For the **zvselh** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions. For the **evsel** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 1721. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	bcctr, bcctrl, bclr, bclrl
Not taken direct branches	0	b, ba, bc, bca, bla, bcla, bl, bcl
Taken direct branches	1	b, ba, bc, bca, bla, bcla, bl, bcl⁽¹⁾
zvselh instruction	00,01,10, or 11	zvselh
evsel instruction	00,01,10, or 11	evsel

1. If the EVCODE for direct branch function calls is not masked in DC4, taken bl and bcl instructions will generate Program Correlation Messages and will not be logged in the history buffer.

75.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that

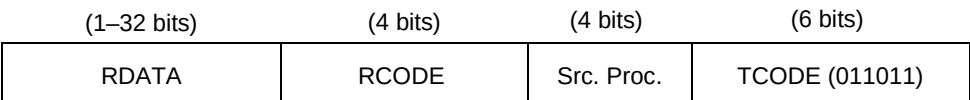
the preceding instruction count and history information should be interpreted in a VLE context.

75.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow; or, if program trace is in history mode, since the last instruction that recorded history information.



Max length = 46 bits; Min length = 15 bits

Figure 1857. Resource Full Message format

Table 1722 shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 1722. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch History. This type of packet is terminated by a stop bit set to 1 after the last history bit.



75.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a “Branch and Link” instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to $MSR_{PMM}=0$ and $DC4_{PTMARK}=1$.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to [Table 1695](#) for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

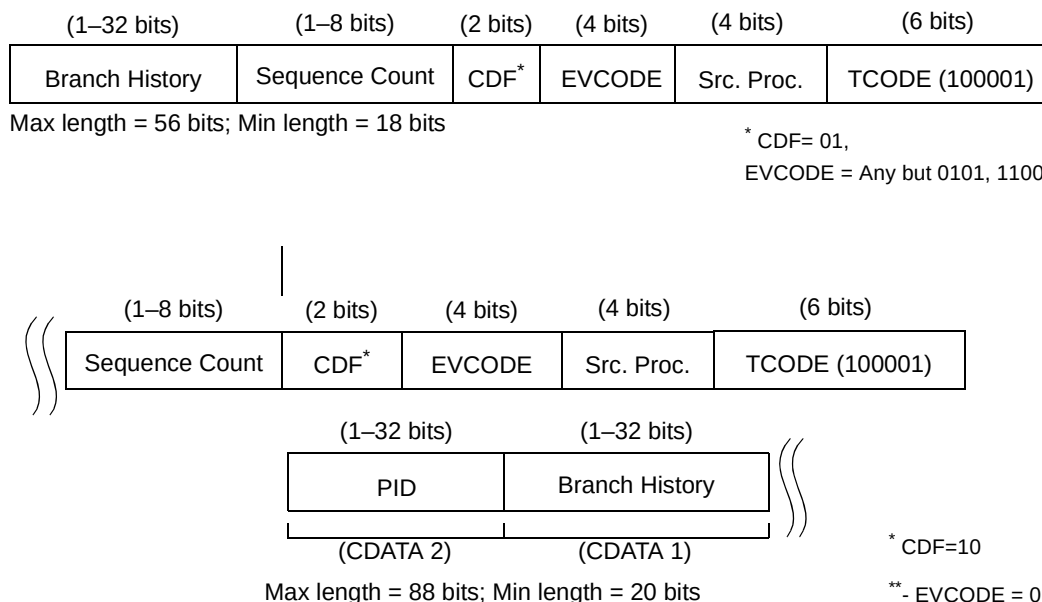


Figure 1858. Program Correlation message formats

75.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

75.12.6 Program Trace Overflow Error messages

an Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

75.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see [Table 1723](#)):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow '1'$ with $DC4_{PTMARK} = '1'$.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Figure 1859. Program Trace Synchronization message format

Exception conditions that result in Program Trace Synchronization are summarized in [Table 1723](#).

Table 1723. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (j_trst_b), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (p_reset_b), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a nex_evti_b assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

75.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see [Table 1723](#)):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync messages is as follows:

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

Figure 1860. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. Messages is as follows:

(1–32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Figure 1861. Indirect Branch History w/ Sync message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in [Table 1724](#).

Table 1724. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync. Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the <i>next</i> Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCMPIDMSG → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

75.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of DC4_{PTMARK}

75.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

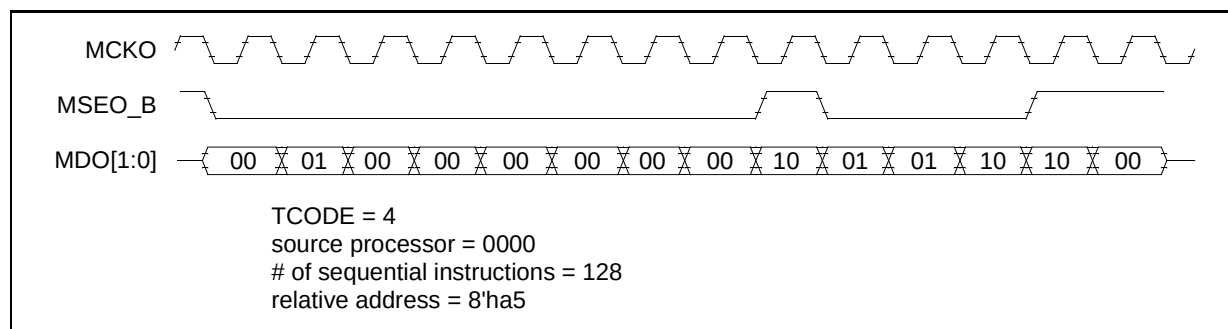


Figure 1862. Program Trace — Indirect Branch message (Traditional)

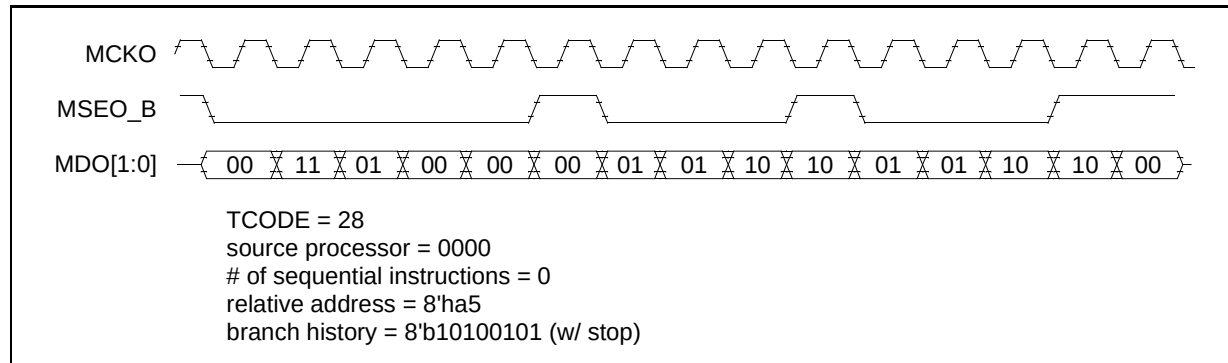


Figure 1863. Program Trace — Indirect Branch message (history)

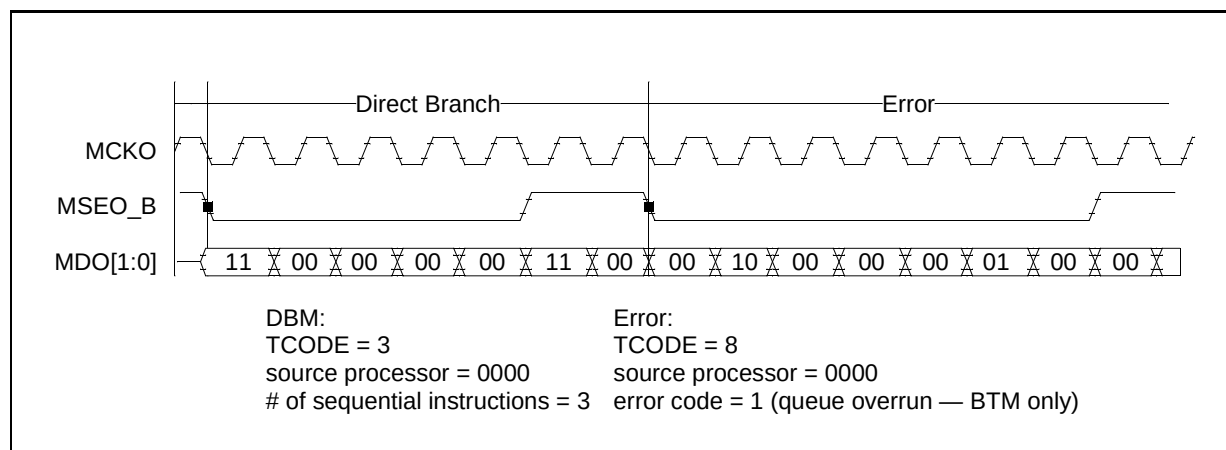


Figure 1864. Program Trace — Direct Branch (traditional) & Error messages

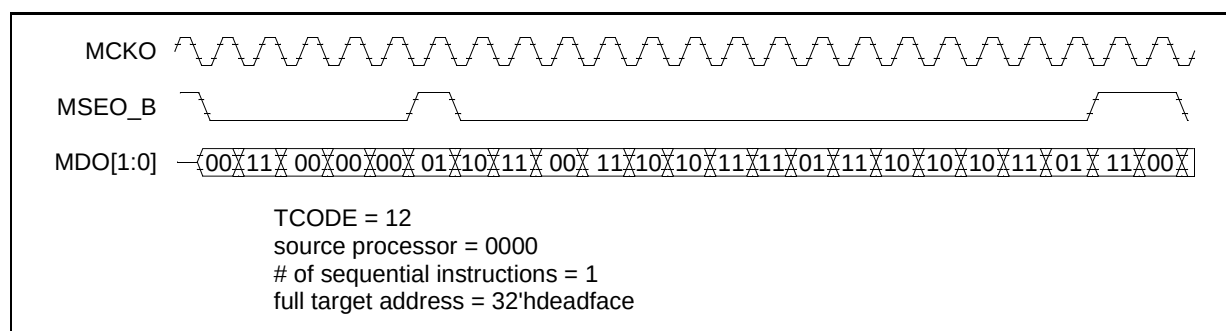


Figure 1865. Program Trace — Indirect Branch w/ Sync message

75.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

75.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note: Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z425n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**)

75.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization, and Error Messages.

75.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace message. Data Write message information is messaged out in the following format:

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 1866. Data Write message format

75.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 1867. Data Read message format

Note: . * Core (e200z425n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Core (e200z425n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

75.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see [Table 1725](#)):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

Figure 1868. Data Write/Read with Sync message format

Exception conditions that result in Data Trace Synchronization are summarized in [Table 1725](#).

Table 1725. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (j_trst_b), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.

Table 1725. Data Trace Exception summary (continued)

Exception Condition	Exception Handling
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a nex_evti_b assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCMPIDMSG → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

75.13.3 DTM operation

75.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

75.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

75.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and DC4_{DTMARK}.

75.13.3.4 Bus cycle special cases

Table 1726. Bus cycle special cases

Special case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{\text{TEA}}$) ⁽¹⁾	Data Trace Message discarded
Bus cycle completed without error ⁽¹⁾	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored

Table 1726. Bus cycle special cases (continued)

Special case	Action
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC _{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) — both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note: For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0, or unless the misaligned access wraps over the end of a circular buffer when using the LSP specialized load or store with circular addressing. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z425n3) Core Debug Support chapter.)

Note: A store to the store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

75.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

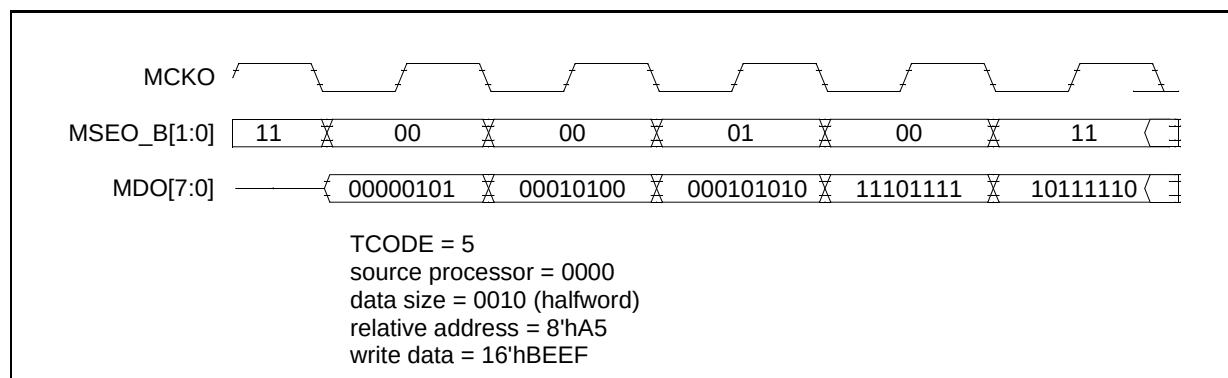


Figure 1869. Data Trace — Data Write message

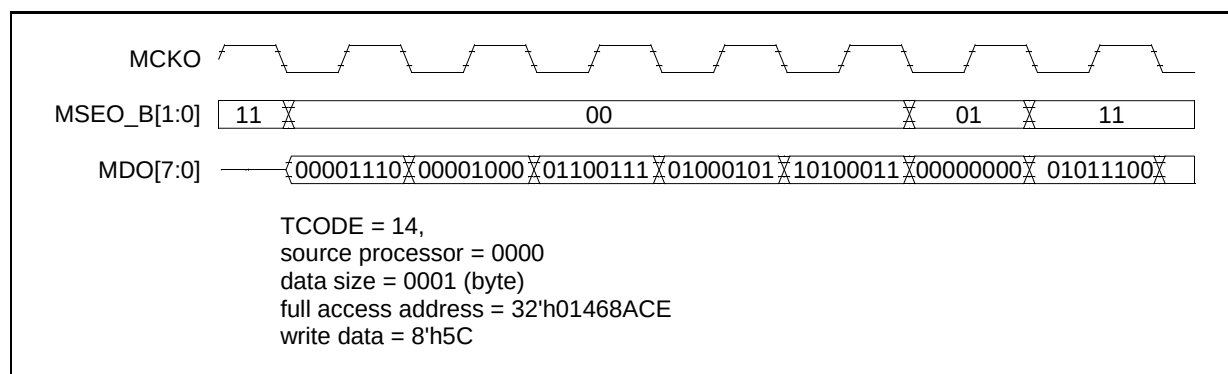


Figure 1870. Data Trace — Data Read w/ Sync message

75.14 Data Acquisition messaging

this section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

75.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

75.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

75.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the “External Debug Resource Allocation Control Register (EDBRAC0)” section in the Core (e200z425n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

(1–32 bits)	(8 bits)	(4 bits)	(6 bits)
DQDATA	DQTAG	Src. Proc.	TCODE (011011)

Max length = 50 bits; Min length = 19 bits

Figure 1871. Data Acquisition message format

75.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z425n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the WMSK_{WEM} field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See [Table 1730](#) for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

(1–30 bits)	(4 bits)	(6 bits)
Watchpoint Source	Src. Proc.	TCODE (001111)

Variable length = 11–40 bits

Figure 1872. Watchpoint message format

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 1727. Watchpoint source encoding

Watchpoint source (1–30 bits)	Watchpoint description
00000000000000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1	Watchpoint #0 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x	Watchpoint #1 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx	Watchpoint #2 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx	Watchpoint #3 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx	Watchpoint #4 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx	Watchpoint #5 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx	Watchpoint #6 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #7 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #8 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #9 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #10 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #11 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #12 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #13 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #14 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #15 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #16 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #17 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #18 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #19 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #20 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #21 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #22 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #23 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #24 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #25 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #26 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #27 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #28 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #29 enabled for WTM

75.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

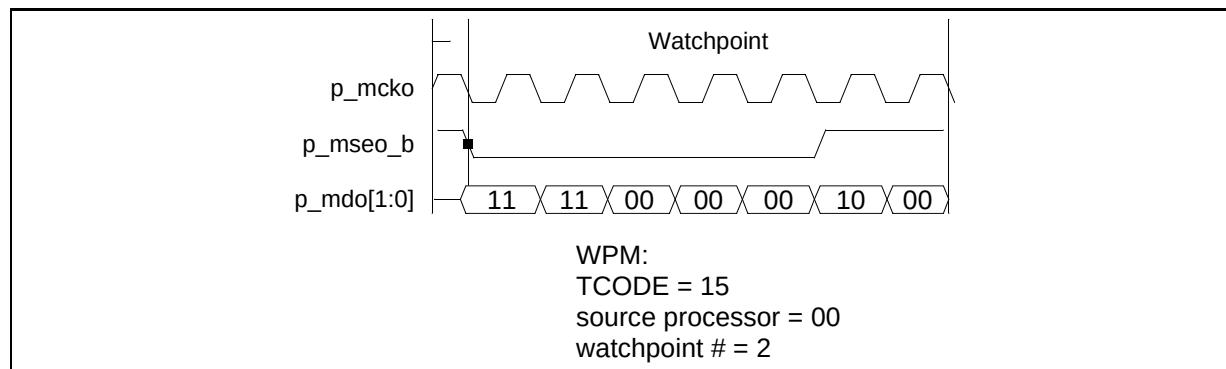


Figure 1873. Watchpoint Message & Watchpoint Error message

75.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a “window” for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect.

[Table 1728](#) details the external hardware trigger signals.

Table 1728. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

Table 1728. External Hardware Trigger Signals (continued)

Name	Description
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0→1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

75.17 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The RWCS_{ATTR} field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note: Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

75.17.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure as follows:
 - a) Write Address → 32h'xxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure the bits as follows:
 - a) Access Control (AC) → 1b'1 (to indicate start access)
 - b) Map Select (MAP) → 3b'000 (primary memory map)
 - c) Access Priority (PR) → 2b'11 (highest priority)
 - d) Read/Write (RW) → 1b'1 (write access)
 - e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
 - f) Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure as follows:
 - a) Write Data → 32h'xxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see [Table 1730](#) for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note: Only the **nex_ahb_start**, **nex_rdy_b**, and **nex_err_b** pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

75.17.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in [Section 75.17.1, Single write access](#) to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has been cleared,

this indicates that the device is ready for the next access in the block transfer, or an error has occurred.

3. If AC has not been cleared due to an error, repeat Step 3 in [Section 75.17.1, Single write access](#) until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

75.17.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure as follows:
 - a) Read Address → 32h'xxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#). Configure the bits as follows:
 - a) Access Control (AC) → 1b'1 (to indicate start access)
 - b) Map Select (MAP) → 3b'000 (primary memory map)
 - c) Access Priority (PR) → 2b'11 (highest priority)
 - d) Read/Write (RW) → 1b'0 (read access)
 - e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
 - f) Access Count (CNT) → 14h'0000 or 14h'0001 (single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see [Table 1730](#) for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#).

Note: Only the **nex_ahb_start**, **nex_rdy_b**, and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

75.17.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in [Section 75.17.3, Single read access](#) to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and

asserts the **nex_rdy_b** pin (see [Table 1730](#) for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.

3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS Register, and asserts the **nex_rdy_b** pin (see [Table 1730](#) for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#).
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in [Section 75.17.3, Single read access](#) until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note: The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

75.17.5 Error handling

The Nexus 3 module handles various error conditions as follows:

75.17.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared).
2. The ERR bit in the RWCS Register is set.
3. The Error Message is sent (TCODE = 8) indicating Read/Write error.

75.17.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
 - a) If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
 - b) If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

75.17.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

(5 bits)	(4 bits)	(6 bits)
Error Code (00011)	Src. Proc.	TCODE (001000)

Fixed length = 15 bits

Figure 1874. Error message format

75.18 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

75.18.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in [Table 1730](#). The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see [Section 75.5, Nexus 3 Register Access via JTAG/OnCE](#) for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the “JTAG/OnCE Pins” section of the Core (e200z425n3) Core Debug Support chapter.

Table 1729. JTAG pins for Nexus 3

JTAG pins	Input/ Output	Description of JTAG pins (included in Nexus 1)
j_tdo	O	The Test Data Output (j_tdo) pin is the serial output for test instructions and data. j_tdo is three-stateable and is actively driven in the “Shift-IR” and “Shift-DR” controller states. j_tdo changes on the falling edge of j_tclk .
j_tdi	I	The Test Data Input (j_tdi) pin receives serial test instruction and data. TDI is sampled on the rising edge of j_tclk .
j_tms	I	The Test Mode Select (j_tms) input pin is used to sequence the OnCE controller state machine. j_tms is sampled on the rising edge of j_tclk .
j_tclk	I	The Test Clock (j_tclk) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (j_trst_b) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in [Table 1730](#).

Table 1730. Nexus 3 auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (nex_mcko) is a free running output clock to development tools for timing of nex_mdo[n:0] & nex_mseo_b[1:0] pin functions. nex_mcko is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (nex_mdo[n:0]) are output pins used for OTM, BTM, and DTM. External latching of nex_mdo[n:0] shall occur on the rising edge of the Nexus3 clock (nex_mcko).
nex_mseo_b[1:0]	O	Message Start/End Out (nex_mseo_b[1:0]) are output pins that indicate when a message on the nex_mdo[n:0] pins has started, when a variable length packet has ended, and when the message has ended. External latching of nex_mseo_b[1:0] shall occur on the rising edge of the Nexus3 clock (nex_mcko). One or two pin MSEO functionality is determined at integration time per SOC implementation
nex_rdy_b	O	Ready (nex_rdy_b) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, nex_rdy_b remains deasserted
nex_err_b	O	Error (nex_err_b) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, nex_err_b remains deasserted
nex_evto_b	O	Event Out (nex_evto_b) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. nex_evto_b is held asserted for one (1) cycle of nex_mcko : 1) One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2) Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01

Table 1730. Nexus 3 auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_evti_b	I	Event In (nex_evti_b) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1) Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2) Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (nex_wevto[3:0]) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. nex_wevto[3:0] is held asserted for one (1) cycle of nex_mcko .
nex_ext_src_id[0:3]	I	nex_ext_src_id[0:3] is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time
nex_sfwcntl_en]	I	nex_sfwcntl_en is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to [Section 75.20, Auxiliary port arbitration](#), for detail on Nexus port arbitration.

Table 1731. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (nex_aux_req[1:0]) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 1733 .)
nex_aux_busy	O	Nexus Auxiliary Busy (nex_aux_busy) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant (nar_aux_grant) is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (ext_multi_nex_sel) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

75.18.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in [Table 1732](#). Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the SPC57EM80xx.

Table 1732. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 1875 illustrates the state diagram for single pin MSEO transfers.

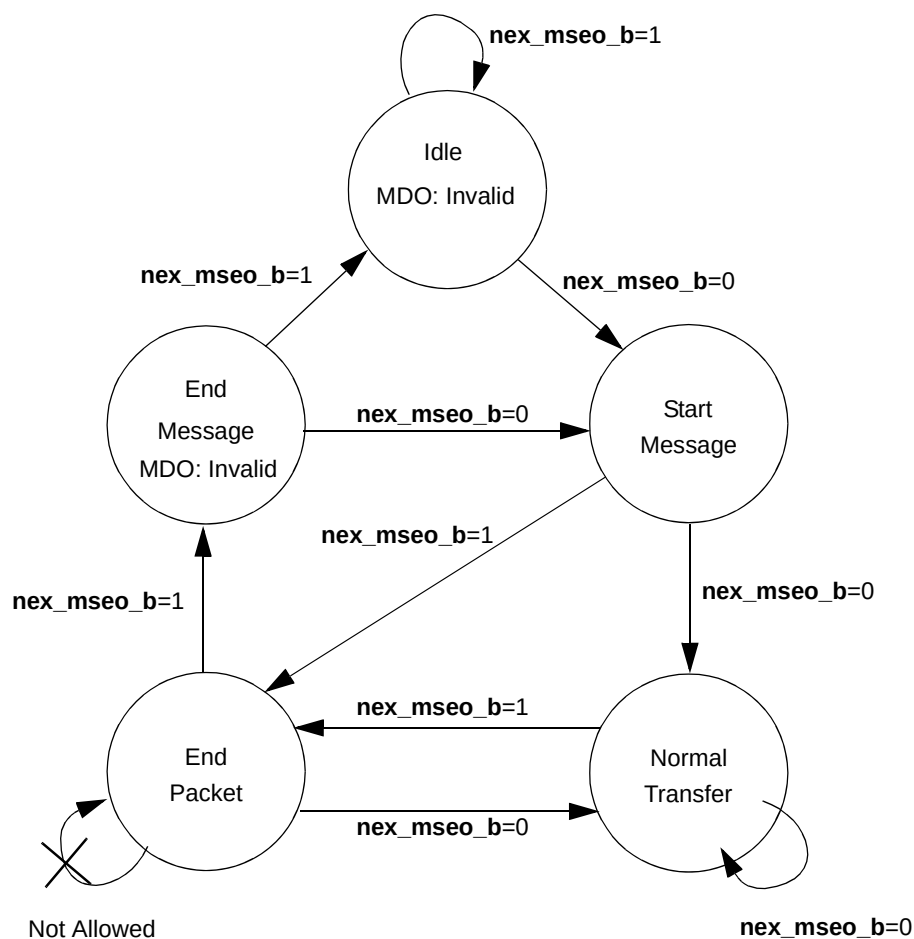


Figure 1875. Single pin MSEO transfers

Note that the “End Message” state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive “End Packet” messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive ‘1’s on the **nex_mseo_b** pin before the actual end of message.

Figure 1876 illustrates the state diagram for dual pin MSEO transfers.

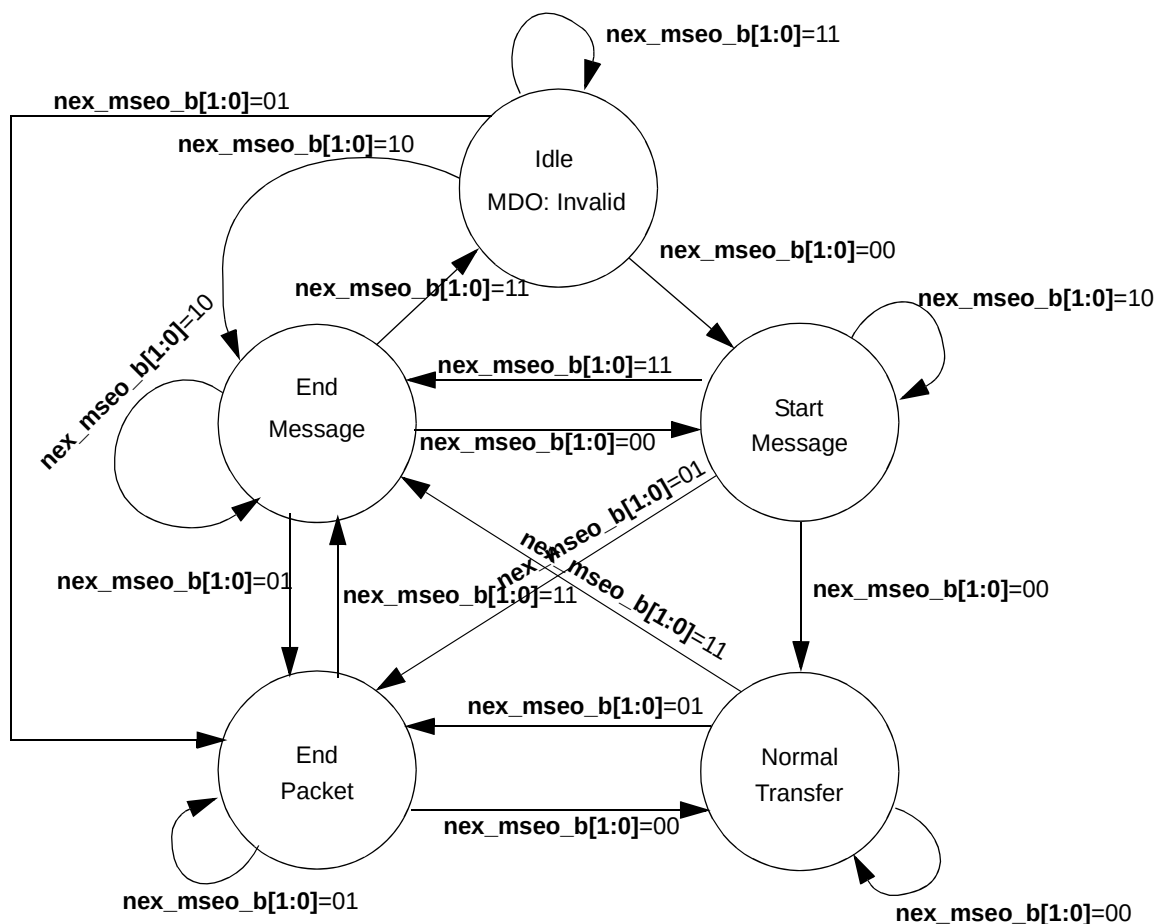


Figure 1876. Dual pin MSEO transfers

The dual pin MSEO option is more robust than the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive “End Packet” states. This can be an advantage when small, variable sized packets are transferred.

Note: The “End Message” state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

75.19 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

75.20 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level. The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in [Table 1733](#) below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in [Section 75.18.2, Pin protocol](#). The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the MSEO state machine reaches the “End Message” state.

Table 1733. MDO request encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

75.21 Examples

The following are examples of Program Trace and Data Trace Messages.

[Table 1734](#) illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. [Table 1735](#) illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- lx = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 1734. Indirect branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 1735. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A P	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

[Table 1736](#) and [Table 1737](#) illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 1736. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 1737. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

[Table 1738](#) illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and [Table 1739](#) illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 1738. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)

Table 1738. Data write message example (8 MDO / 1 MSEO) (continued)

Clock	nex_mdo[7:0]									nex_mseo_b	State
1	S1	S0	T5	T4	T3	T2	T1	T0		0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2		1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0		0	Normal Transfer
4	0	0	0	0	0	0	0	0		1	End Packet
5	0	0	0	0	0	0	0	0		1	End Message

Table 1739. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]									nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0		0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2		0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0		1	1	End Packet/ End Message

75.22 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the SPC57EM80xx Data Sheet.

75.23 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

75.23.1 JTAG sequence for accessing internal Nexus registers

Table 1740. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE → SELECT-DR_SCAN
2	0	SELECT-DR_SCAN → CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR → SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR → EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR → UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR → SELECT-DR_SCAN
8	0	SELECT-DR_SCAN → CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR → SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value

Table 1740. Accessing internal Nexus 3 registers via JTAG/OnCE (continued)

Step #	TMS Pin	Description
11	1	SHIFT-DR → EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR → UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR → RUN-TEST/IDLE (transfer complete — Nexus controller to Reg. Select state)

75.23.2 JTAG sequence for read access of memory-mapped resources

Table 1741. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of nex_rdy_b or nex_err_b pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

75.23.3 JTAG sequence for write access of memory-mapped resources

Table 1742. Accessing memory-mapped resources (writes)

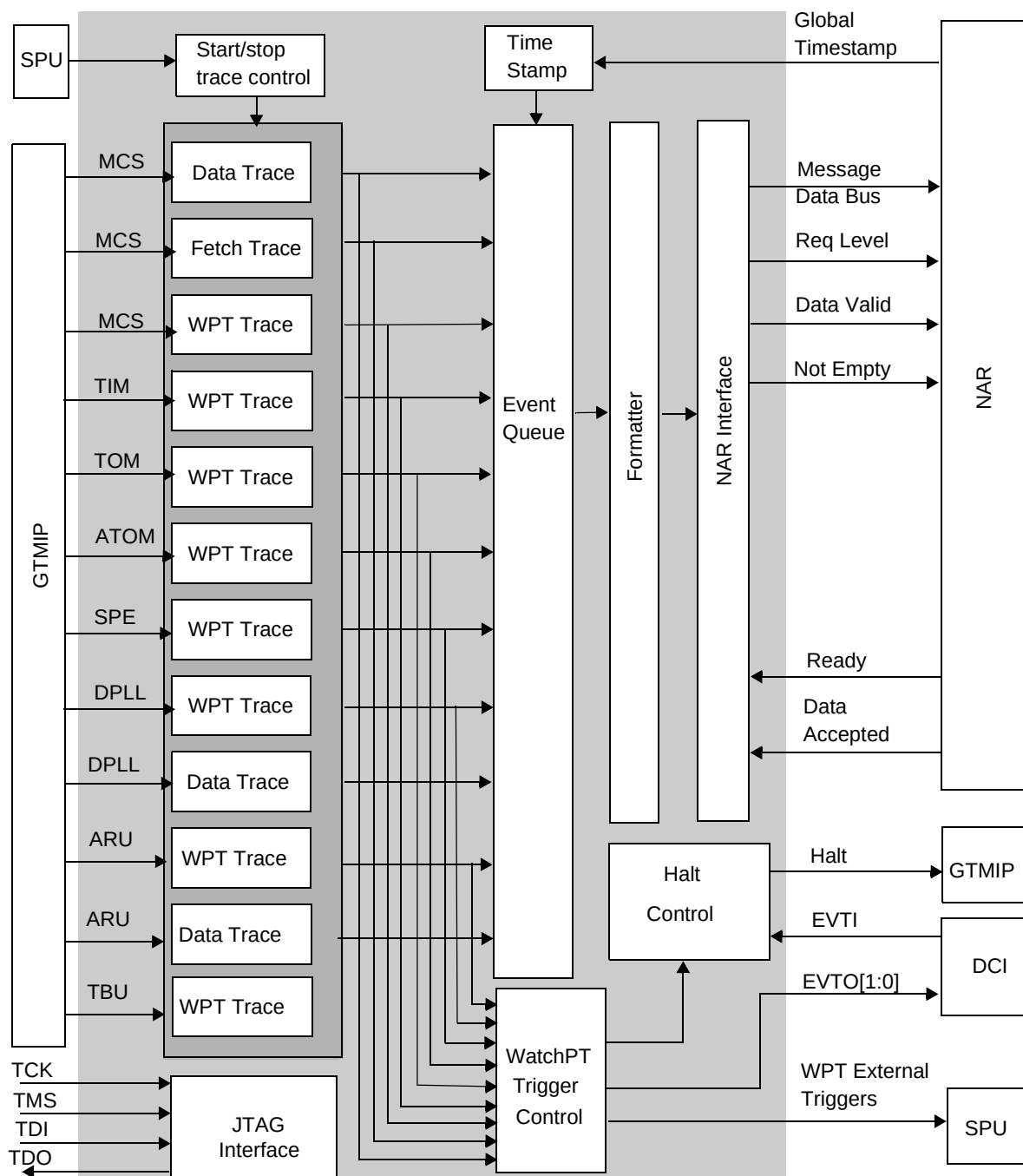
Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value — data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address — data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of nex_rdy_b or nex_err_b pin
8	—	If CNT > 0, go back to Step #5

76 GTM Development Interface (GTMDI)

76.1 Introduction

Figure 1877 shows a block diagram of the Generic Timer Module (GTM) Development Interface.

Figure 1877. GTMDI block diagram



76.2 Overview

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including Multi-Channel Sequencer (MCS) cores, Advanced Routing Unit (ARU) data trace, input channel filters trace, output timer channels, DPLL and Sensor Pattern Evaluation (SPE) sub-modules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules to implement complex debug features. Trace capability is also provided along with a global time-stamp which allows real time trace for several GTM sub-module events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and NAR on-chip modules. The Nexus standard is only used for the physical interfaces. Due to specific requirements for the GTM debug, the Nexus standard is not followed.

[Figure 1878](#) describes the interconnection between the GTMDI and GTM modules. The GTMDI selects which signal or bus to monitor since it does not have the capability to monitor all signals in parallel. The multiplexing architecture is described in the following sections.

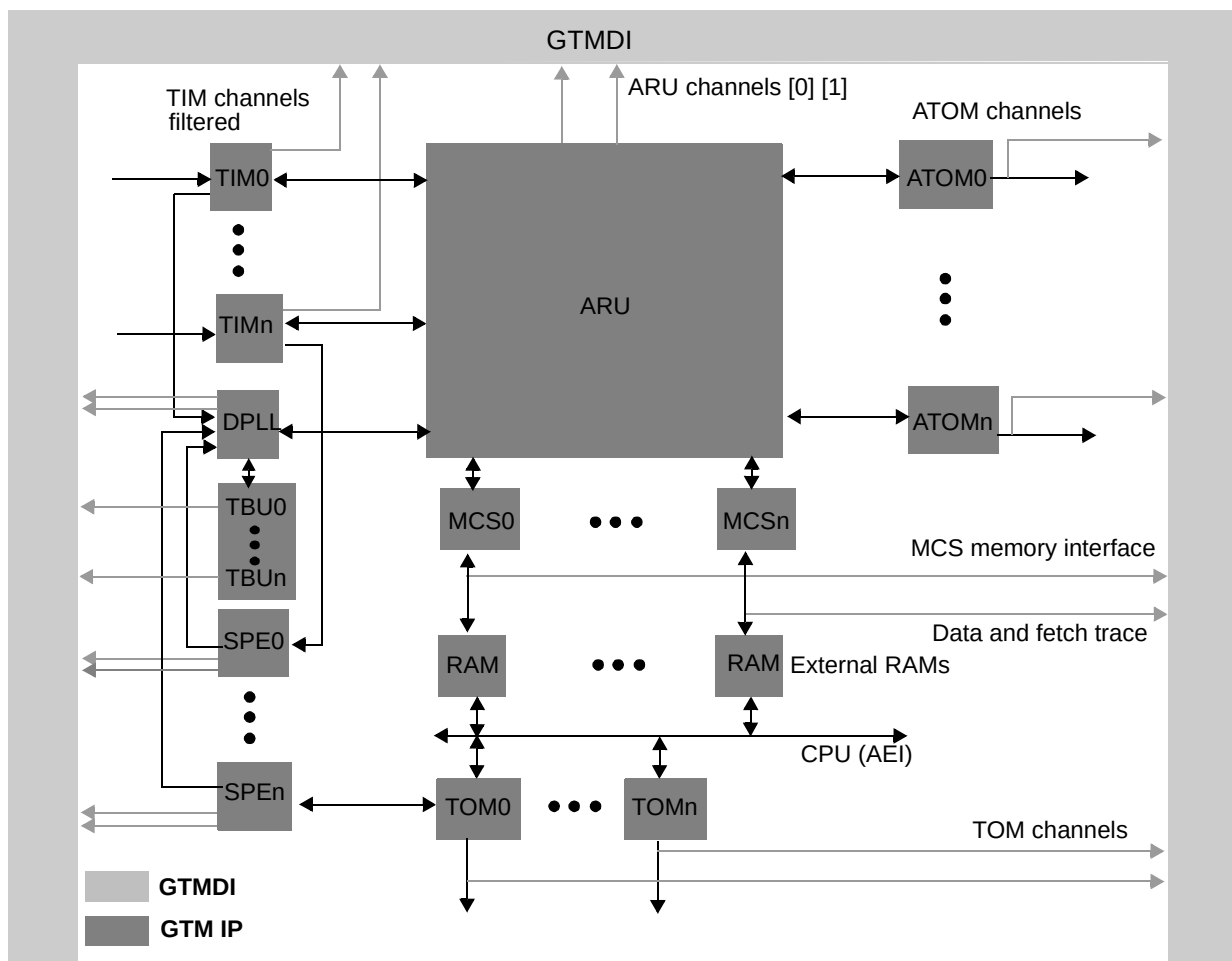


Figure 1878. GTMDI and GTM interface

Note: [Figure 1878](#) does not show all of the GTM internal modules. It shows only the modules that interface to the GTMDI debug module.

The number of ATOM, TIM, TOM, MCS and SPE modules in [Figure 1878](#) is shown in [Table 1743](#).

Table 1743. GTM Sub-module instances

Module	Number of instances
ATOM	5
TIM	4
TOM	3
MCS	numMCS
SPE	2

[Figure 1879](#) shows the Watchpoint Triggers (WPT) and Watchpoint Messages (WPM). Each GTM sub-module generates two watchpoints which can be routed internally to the GTMDI and generate watchpoint triggers or routed to the message formatter and generate watchpoint messages. The selection between watchpoint triggers and messages is implemented per sub-module. Messages and watchpoint triggers can also be generated at the same time based on the same event within a sub-module, thus a TIM channel transition can generate a watchpoint trigger and a watchpoint message. The watchpoint triggers may be sent to the Sequence Processing Unit (SPU) outside the GTMDI if enabled by appropriate control register bits for each sub-module, such as Timer Input Module (TIM), Timer Output Module (TOM) and ARU-connected Timer Output Module (ATOM). The watchpoint messages, if enabled, are sent to the NAR through the Nexus Client interface.

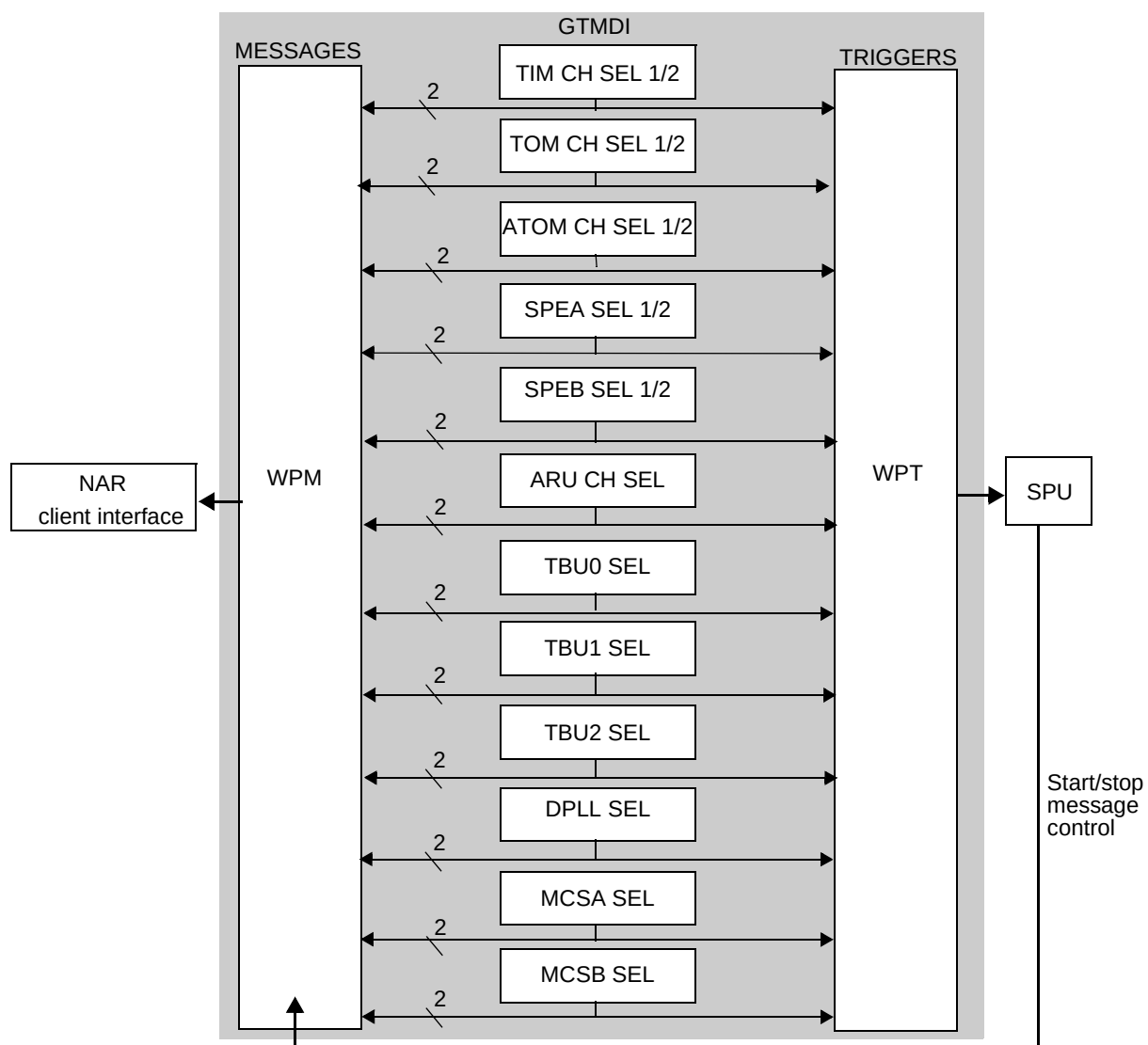


Figure 1879. Watchpoint triggers and watchpoint messages

The SPU provides external control of the watchpoint messages generated by the GTMDI thus controlling the bandwidth over the Nexus Client interface. This functionality is implemented through Start/Stop control signals from the SPU. These signals are connected to the sub-module register bit that enables the message transmission implementing a dual control over these registers, either by JTAG interface programming or by SPU Start/Stop signals. See [Section 76.5, Functional description](#) for more details about register control and general architecture.

76.2.1 Features

The GTMDI block implements the following features:

- Full duplex pin interface for medium and high visibility throughput
 - 2 $\overline{\text{EVTO}}$ (Event Out) for GTM real time signal monitoring
 - 1 $\overline{\text{EVTI}}$ (Event In)
 - IEEE 1149.1 (JTAG) Test Access Port (TAP)
 - 4 pins (TDI, TDO, TMS, and TCK)
 - Reset input $\overline{\text{TRST}}$ driven by either the Nexus Port Controller or an external pin
- GTM Development Support
 - Read/Write registers in debug mode from JTAG port
 - Ability to enter debug mode at reset negation or during normal execution

Note: Enter debug out-of-reset is provided through the SoC system level debug control signal. In order to allow this functionality the GDE and SDBE control bits in the GTMDI_DC register should be set.

- Halt and Watchpoint Configuration
- MCS Data Trace Messaging (DTM) and Watchpoint Messaging (WPM) allows the development tool to trace reads and writes to selected MCS RAM address ranges
 - Two data trace windows for each MCS core with programmable address ranges and access attributes are provided. Data trace windowing reduces the requirement on the Aurora port bandwidth by constraining the number of trace locations.
 - Fetch trace capability controlled by two programmable Watchpoints.
 - Private messages are used to indicate special MCS core cases not covered by public messages. Two MCS cores can be debugged at the same time.

Note: Enabling two MCS cores at the same time for data trace can exceed the capacity of the NAR Client interface throughput thus information can be lost.

- DPLL Data Trace Messaging and Watchpoint Trace Messaging allows the development tool to trace reads and writes to selected DPLL RAM addresses and selected data value.
- ARU Data Trace Messaging and Watchpoint Trace Messaging allows the development tool to trace selected ARU channels.
- Watchpoint Messaging (WPM)
- Trigger Control
 - The GTMDI provides selection of trigger signals that generate debug actions at system level such as Halt GTM itself or cores in the same device.
 - Trace enable signals are provided to control the start/stop of trace messages for each one of the GTM Nexus message sources, such as Timer Input Module (TIM), Timer Output Module (TOM), ARU-connected Timer Output Module (ATOM), etc.
- GTMDI clock operation
 - The module is able to operate in any frequency ratio considering GTMDI and JTAG frequency. The GTMDI operates from the same clock source as the GTM IP module.

76.2.2 Modes of operation

The GTMDI block is placed in reset when the Nexus reset signal is asserted. The JTAG registers controlled by the TCK clock are also put into the reset state if the TAP controller is in the TEST-LOGIC-RESET state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the TEST-LOGIC-RESET state regardless of the initial TAP controller state. Asserting the Nexus reset signal results in asynchronous entry into the reset state not only of the TCK controlled registers but all registers in the GTMDI. The GTMDI is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the TEST-LOGIC-RESET state
- The auxiliary output port pins are negated
- The auxiliary output port enable outputs are negated
- The TDI, TMS, and TCK TAP inputs are ignored (The TMS and TCK signal are ignored only while the Nexus reset signal is asserted.)
- Registers default back to their reset values

Note: The GTMDI is not reset by the system reset signal. It is reset during device Power On reset or by a dedicated Nexus reset thus preserving internal registers state and programming during system reset.

Debug Mode can be controlled by on-chip logic through the system debug signal. The level on this signal defines if the GTM module is in debug mode, meaning that it is in Halt state and accepts write and read accesses to internal registers that do not modify the state of those registers. If asserted, the GTM enters Halt state. If negated, the GTM debug mode is controlled by other sources, internal to the GTMDI. The GTM may enter debug out of reset by asserting the system debug signal while the device is being reset. By detecting this condition, the GTMDI puts the GTM into the Halt state. The GDE and SDBE bits described in [Figure 1881](#) must be set during reset allowing the GTM to enter debug state.

76.2.3 DATA port

GTMDI messages are sent through the Message Data Bus. The Message Data Bus has a fixed width of 32 bits which defines one BEAT of data. There are three BEATS per NAR clock thus 96 bits are transmitted in parallel.

76.3 External signal description

The GTMDI pinout provides interface for the transmission of messages and for accessing Nexus registers. The GTMDI pin definition is outlined in [Table 1744](#).

Table 1744. GTMDI signal properties

Name	Port	Function	Reset State
EVTI	Auxiliary	Event In pin	—
$\overline{\text{EVTO}}[1:0]$	Auxiliary	Event Out pin	[11]
TCK	TAP	Test Clock Input	—
TDI	TAP	Test Data Input	—
TDO	TAP	Test Data Output	0
TMS	TAP	Test Mode Select Input	—
$\overline{\text{TRST}}$	TAP	Test Reset Input	—

Each of the signals listed in [Table 1744](#) is described in more detail in the following sections.

76.3.1 Event in ($\overline{\text{EVTI}}$)

Event In ($\overline{\text{EVTI}}$) is used to generate a Halt if enabled by the corresponding control bit. $\overline{\text{EVTI}}$ is an edge-sensitive signal. It is synchronized at the GTMDI input and a falling edge is detected.

76.3.2 Event out $\overline{\text{EVTO}}[1:0]$

Event Out ($\overline{\text{EVTO}}$) are two outputs from GTMDI for development tools with exact timing for a two watchpoints or Halt occurrence. $\overline{\text{EVTO}}$ is asserted for one system clock cycle when Halt or watchpoints are monitored. When internal GTM signals are monitored, such as TIM filtered signals or TOM/ATOM outputs, the $\overline{\text{EVTO}}$ represent the exact signal shape with a certain delay. In this monitoring mode the signal is not inverted through the $\overline{\text{EVTO}}$ output. The real time monitoring mode is intended for oscilloscope type of monitoring thus allowing the user to evaluate the actual shape of the signals. The source selection for the $\overline{\text{EVTO}}$ is centralized in the Development Control register GTMDI_DC.

76.3.3 Test clock input (TCK)

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the JTAG port.

76.3.4 Test data input (TDI)

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

76.3.5 Test data output (TDO)

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is actively driven in the SHIFT-IR and SHIFT-DR states of the JTAG state machine. TDO changes on the falling edge of TCK and is sampled on the rising edge of TCK.

76.3.6 Test Mode Select (TMS)

Test Mode Select (TMS) is an input pin used to sequence the JTAG state machine. TMS is sampled on the rising edge of TCK.

76.4 Register definition

This section provides a detailed description of all GTMDI registers accessible to the development tool. Individual bit-level descriptions and reset states of each register are included. [Table 1745](#) shows the GTMDI registers and index values. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 1745. GTMDI registers

Register	Index	Read/write
Device Identity (DID)	0	R
Reserved	1–4	—
GTMDI Development Control (GTMDI_DC)	5	R/W
GTMDI Development Status (GTMDI_DS)	6	R
Reserved	7–10	—
TIM Watchpoint Control 1 (GTMDI_TIM_WPC1)	11	R/W
TIM Watchpoint Control 2 (GTMDI_TIM_WPC2)	12	R/W
Reserved	13–16	—
TOM Watchpoint Control 1 (GTMDI_TOM_WPC1)	17	R/W
TOM Watchpoint Control 2 (GTMDI_TOM_WPC2)	18	R/W
Reserved	19–22	—
ATOM Watchpoint Control 1 (GTMDI_ATOM_WPC1)	23	R/W
ATOM Watchpoint Control 2 (GTMDI_ATOM_WPC2)	24	R/W
Reserved	25–28	—
SPEA Watchpoint Control 1 (GTMDI_SPEA_WPC1)	29	R/W
SPEA Watchpoint Control 2 (GTMDI_SPEA_WPC2)	30	R/W
Reserved	31–34	—
SPEB Watchpoint Control 1 (GTMDI_SPEB_WPC1)	35	R/W
SPEB Watchpoint Control 2 (GTMDI_SPEB_WPC2)	36	R/W
Reserved	37–40	—
DPLL Development Control (GTMDI_DPLL_WPC1)	41	R/W
DPLL Watchpoint Control 1 (GTMDI_DPLL_WPC2)	42	R/W
DPLL Watchpoint Control 2 (GTMDI_DPLL_WPC3)	43	R/W
DPLL Watchpoint Control 3 (GTMDI_DPLL_WPC4)	44	R/W
DPLL Watchpoint Control 4 (GTMDI_DPLL_WPC5)	45	R/W
DPLL Data Trace Control (GTMDI_DPLL_DTC)	46	R/W

Table 1745. GTMDI registers (continued)

Register	Index	Read/write
Reserved	47–49	—
ARU Watchpoint Control (GTMDI_ARU_WPC1)	50	R/W
ARU Watchpoint Control (GTMDI_ARU_WPC2)	51	R/W
ARU DATA0H (GTMDI_ARU_DATA0H)	52	R/W
ARU DATA0L (GTMDI_ARU_DATA0L)	53	R/W
ARU DATA1H (GTMDI_ARU_DATA1H)	54	R/W
ARU DATA1L (GTMDI_ARU_DATA1L)	55	R/W
ARU Data Trace Control (GTMDI_ARU_DTC)	56	R/W
Reserved	57–59	—
MCSA Development Control (GTMDI_MCSA_DC)	60	R/W
Reserved	61	—
MCSA Watchpoint Control (GTMDI_MCSA_WPC)	62	R/W
MCSA Program Fetch Trace Control (GTMDI_MCSA_PTC)	63	R/W
MCSA Data Trace Control (GTMDI_MCSA_DTC)	64	R/W
MCSA Watchpoint Address 1 (GTMDI_MCSA_WPA1)	65	R/W
MCSA Watchpoint Address 2 (GTMDI_MCSA_WPA2)	66	R/W
MCSA Watchpoint Data 1 (GTMDI_MCSA_WPD1)	67	R/W
MCSA Watchpoint Data 2 (GTMDI_MCSA_WPD2)	68	R/W
MCSA Program Trace Channel Enable (GTMDI_MCSA_CE)	69	R/W
Reserved	70	—
MCSA Data Trace Address Range 1 (GTMDI_MCSA_DTAR1)	71	R/W
MCSA Data Trace Address Range 2 (GTMDI_MCSA_DTAR2)	72	R/W
Reserved	73–76	—
MCSB Development Control (GTMDI_MCSB_DC)	77	R/W
Reserved	78	—
MCSB Watchpoint Control (GTMDI_MCSB_WPC)	79	R/W
MCSB Program Fetch Control (GTMDI_MCSB_PTC)	80	R/W
MCSB Data Trace Control (GTMDI_MCSB_DTC)	81	R/W
MCSB Watchpoint Address 1 (GTMDI_MCSB_WPA1)	82	R/W
MCSB Watchpoint Address 2 (GTMDI_MCSB_WPA2)	83	R/W
MCSB Watchpoint Data 1 (GTMDI_MCSB_WPD1)	84	R/W
MCSB Watchpoint Data 2 (GTMDI_MCSB_WPD2)	85	R/W
MCSB Program Trace Channel Enable (GTMDI_MCSB_CE)	86	R/W
Reserved	87	—
MCSB Data Trace Address Range 1 (GTMDI_MCSB_DTAR1)	88	R/W

Table 1745. GTMDI registers (continued)

Register	Index	Read/write
MCSB Data Trace Address Range 2 (GTMDI_MCSB_DTAR2)	89	R/W
Reserved	90–93	—
TBU0 Watchpoint Control 1 (GTMDI_TBU0_WPC1)	94	R/W
TBU0 Watchpoint Control 2 (GTMDI_TBU0_WPC2)	95	R/W
TBU0 DATA (GTMDI_TBU0_DATA)	96	R/W
Reserved	97–100	—
TBU1 Watchpoint Control 1 (GTMDI_TBU1_WPC1)	101	R/W
TBU1 Watchpoint Control 2 (GTMDI_TBU1_WPC2)	102	R/W
TBU1 DATA (GTMDI_TBU1_DATA)	103	R/W
Reserved	104–107	—
TBU2 Watchpoint Control 1 (GTMDI_TBU2_WPC1)	108	R/W
TBU2 Watchpoint Control 2 (GTMDI_TBU2_WPC2)	109	R/W
TBU2 DATA (GTMDI_TBU2_DATA)	110	R/W

76.4.1 Register descriptions

This section consists of GTMDI register descriptions. The registers are shown in [Table 1745](#). The MCS cores selected for debugging are named MCSA and MCSB thus two MCS cores can be debugged at the same time. See registers GTMDI_MCSA_DC and GTMDI_MCSB_DC which specify the real number of the MCS associated with MCSA and MCSB.

76.4.1.1 Device identity register (DID)

The DID register contains the unique identification information as shown in [Figure 1880](#).

Register index: 0

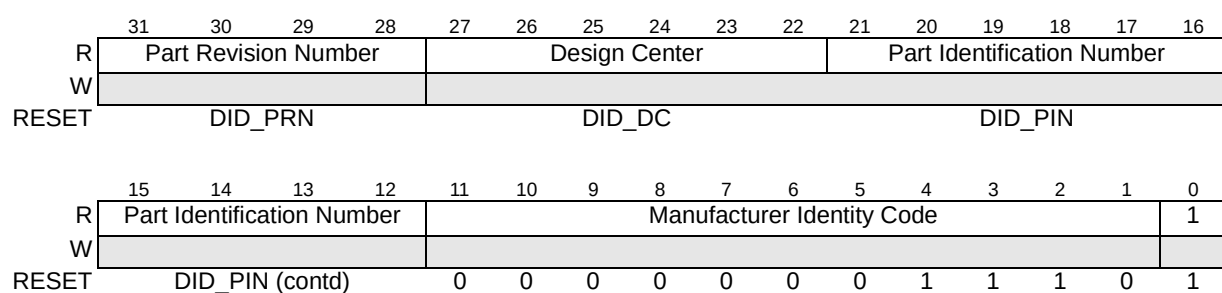


Figure 1880. Device identity register (DID)

[Table 1746](#) describes the device identification register functions.

Table 1746. DID field descriptions

Field	Description
31–28 PRN	Part Revision Number. Contains the revision number of the part.
27–22 DC	Design Center. Indicates the design center.
21–12 PIN	Part Identification Number. Contains the part number of the device. Value is 0x3E3.
11–1 MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID.
0 IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

76.4.1.2 GTMDI development control register (GTMDI_DC)

The GTMDI_DC register controls various trace and debug features of the GTM modules as described in [Figure 1881](#).

Register index: 5

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GD	0	0	0	EOS0				0	0	0	0	EOS1			
W	E															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	EIDR	0	DBE	DBR	SDBE	0	TSFT
W								CHR								
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1881. GTMDI development control register (GTMDI_DC)

[Table 1747](#) describes the GTMDI_DC register functions.

Table 1747. GTMDI_DC field descriptions

Field	Description
31 GDE	Global Debug Enable. This control bit disables all debug features of the GTMDI module including ability to put GTM IP in Halt state, modify EVTO output, send messages to the DATA bus and set triggers to the SPU module. Before changing the GTMDI configuration it is recommended to clear GDE bit to prevent spurious behavior of the module outputs. See Figure 1926 for details about debug enable logic. 0 GTMDI Global debug is disabled 1 GTMDI Global debug is enabled
27–24 EOS0	EVTO0 Output Selection. This signal is dedicated to sel 1 which is the selection 1 from the indicated GTM modules. See the control registers for each module for a description of this selection. For TIM, TOM and ATOM selections the input filtered signal or output channel signal are used instead of the triggers. 0000 No event on output EVTO0 remains at one 0001 Halt Indication, If EVTO0 is zero indicates that GTM is halted 0010 TIM, Map EVTO0 from TIM channel filter output using ssel 1 selection control 0011 TOM, Map EVTO0 from TOM channel output using ssel1 selection control 0100 ATOM, Map EVTO0 from ATOM channel output using ssel 1 selection control 0101 SPEA, Map EVTO0 from SPEA, controlled by TSS1 described in Figure 1889 0110 SPEB, Map EVTO0 from SPEB, controlled by TSS1 described in Figure 1891 0111 ARU CH0, Map EVTO0 from ARU CH0 1000 DPLL, Map EVTO0 from DPLL according to TSEL1 in Figure 1893 1001 MCSA, Map EVTO0 from MCSA MCSA_sel selection control 1010 MCSB, Map EVTO0 from MCSB MCSB_sel selection control 1011 TBU0 Watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU0_WPC1 register 1100 TBU1 watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU1_WPC1 register 1101 TBU2 watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU2_WPC1 register 1110 Reserved 1111 Reserved
19–16 EOS1	EVTO1 Output Selection. This signal is dedicated to sel 2 which is the selection 2 from the indicated GTM modules. See the control registers for each module for a description of this selection. For TIM, TOM and ATOM selections the input filtered signal or output channel signal are used instead of the triggers. 0000 No event on output EVTO1 remains at one 0001 Halt Indication, If EVTO1 is zero indicates that GTM is halted 0010 TIM, Map EVTO1 from TIM channel filter output using ssel 1 selection control 0011 TOM, Map EVTO1 from TOM channel output using ssel1 selection control 0100 ATOM, Map EVTO1 from ATOM channel output using ssel 1 selection control 0101 SPEA, Map EVTO1 from SPEA, controlled by TSS2 described in Figure 1889 0110 SPEB, Map EVTO1 from SPEB, controlled by TSS2 described in Figure 1891 assert_spe1_dir_evto 0111 ARU CH1, Map EVTO1 from ARU CH1 1000 DPLL, Map EVTO1 from DPLL according to TSEL2A/B in Figure 1893 1001 MCSA, Map EVTO1 from MCSA MCSA_sel selection control 1010 MCSB, Map EVTO1 from MCSB MCSB_sel selection control 1011 TBU0 Watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU0_WPC1 register 1100 TBU1 watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU1_WPC1 register 1101 TBU2 watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU2_WPC1 register 1110 Reserved 1111 Reserved

Table 1747. GTMDI_DC field descriptions (continued)

Field	Description
8 CHR	Clear Halt Request. Writing this bit to one clears halt requests., which means GTM resumes normal operation out of halt state. The status bits of the GTMDI_DS register are cleared. This bit is self-cleared thus read always as zero. 0 No action 1 Clear halt requests
6 EIDR	EVTI Debug Request enable. The EIDR bit enables EVTI input to request the GTM to enter Halt state. After the GTM enters Halt state due to a pulse in EVTI signal, the EIDR bit state is not meaningful, thus it may be cleared and the GTM continues in Halt state. See Figure 1926 for details about debug enable logic. 0 Disable EVTI to control GTM to enter Halt state 1 Enable EVTI to control GTM to enter Halt state
4 DBE	Debug Enable. DBE enables debug mode thus allowing the GTM to enter debug mode which means halt regular operation. The debug events controlled by this register are the ones generated internally to the GTMDI, excluding ipg_debug and EVTI which have their dedicated enable control. See Figure 1926 for details about debug enable logic. 0 Debug mode disabled 1 Debug mode enabled
3 DBR	Debug Request. Mechanism to allow development tool to request modules to enter debug mode directly. See Figure 1926 for details about debug enable logic. 0 Does not modify GTM IP debug state 1 Request GTM IP to enter debug mode
2 SDBE	System Debug Enable. The SDBE bit enables the GTM to enter halt state if system level debug enable signal is asserted. This bit allows the debug controller to select specific sub-groups of cores or modules to halt, while others continue to operate. GDE bit must also be set to allow GTM to halt. See Figure 1926 for details about debug enable logic. 0 GTM does not enter halt state due to system debug enable signal assertion 1 GTM is enabled to enter halt state due to system debug enable signal assertion
0 TSFT	Timestamp Force Transmission. Forces the transmission of the Timestamp value along with the Nexus message data for all messages transmitted by the GTMDI Client module. When Timestamp inclusion is not forced into the client message, its inclusion depends upon the NAR settings. 0 Does not force transmission of Timestamp value 1 Forces transmission of Timestamp along with Nexus message

76.4.1.3 GTMDI development status register (GTMDI_DS)

The GTMDI_DS register described in [Figure 1882](#) shows the status of various conditions that impact GTM development support. All status bits are dynamic and do not require clearing. Debug Status Event is queued due to a change in the Halt or Stop modes and a Debug Status Message is generated as soon as there is an opportunity in the NAR interface. If any status bit changes before the Status message is sent, this change is reflected in the message. The GTMDI_DS register dynamically tracks the debug condition status even when several of them are simultaneously set.

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HLT	0	0	HS1												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STP	0	0	HS2												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1882. GTMDI development status register (GTMDI_DS)

Table 1748 describes the GTMDI_DS register functions.

Note: The HS1 and HS2 bit fields are cleared as soon as the GTM resumes normal operation which is controlled by the CHR bit in the GTMDI_DC register. One exception is HS1[0] that indicates system debug request, which is not cleared by CHR. This bit is cleared when system debug request is negated.

Table 1748. GTMDI_DS field descriptions

Field	Description
31 HLT	Halt Indication. The HLT bit indicates if the GTM IP is in Halt state, ready to receive debug accesses. 0 GTM is running 1 GTM is in Halt state
28–16 HS1	Halt Status1. The HS1 field shows which source generated the halt condition. More than one source can generate the halt condition at the same time. This register is only meaningful if the GTM is halted. Its content indicates the halt sources that occurred immediately when the halt mode is entered. 0000000000000 No Trigger generated 1xxxxxxxxx TIM x1xxxxxxxxx TOM xx1xxxxxxxxx ATOM xxx1xxxxxxxxx SPEA xxxx1xxxxxxxxx SPEB xxxxx1xxxxxxx ARU xxxxxx1xxxxxx DPLL xxxxxxx1xxxxx MCSA xxxxxxx1xxxxx MCSB xxxxxxx1xxx TBU0 Watchpoint 1 xxxxxxx1xx TBU1 Watchpoint 1 xxxxxxx1x TBU2 Watchpoint 1 xxxxxxx1 External through system debug enable signal

Table 1748. GTMDI_DS field descriptions (continued)

Field	Description
15 STP	Stop Status. Indicates if the GTM is in stop mode with no active clocks. Besides the system level stop mode mechanism, the GTM may independently be programmed for STOP mode by setting the Module Disable Control bit. Thus, the STP field indicates if the GTM is in stop mode no matter if due to the system stop mode assertion or due to the MDIS bit in the GTM module control register. 0 GTM not stopped 1 GTM is stopped in low power mode
12–0 HS2	Halt Status². Similarly to HS1 this field indicates the source for the halt condition selection 2. See Figure 1927 for reference about source selection 1 and 2. The HS2 field shows which source generated the halt condition 2. More than one source can generate the halt condition at the same time. This register is only meaningful if the GTM is halted. Its content indicates the halt condition source that occurred when the halt mode is entered. 00000000000000No Trigger generated 1xxxxxxxxxxx TIM x1xxxxxxxxxxx TOM xx1xxxxxxxxxxx ATOM xxx1xxxxxxxxxxx SPEA xxxx1xxxxxxxxxxx SPEB xxxxx1xxxxxxx ARU xxxxxx1xxxxxxx DPLL xxxxxxx1xxxxxxx MCSA xxxxxxx1xxxxxxx MCSB xxxxxxx1xxx TBU0 Watchpoint 2 xxxxxxx1xx TBU1 Watchpoint 2 xxxxxxx1x TBU2 Watchpoint 2 xxxxxxx1 EVTI

76.4.1.4 TIM watchpoint control register 1 (GTMDI_TIM_WPC1)

The GTMDI_TIM_WPC1 register shown in [Figure 1883](#) controls the Watchpoint Triggers WPT issued by TIM channels. These triggers are used by the GTM internal modules in order to control GTM Halt mode entering and also at SoC level. See [Figure 1927](#) which describes the use of this register.

Note: *Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG. The SEN1 and SEN2 control bits from the GTMDI_TIM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.*

Figure 1883. TIM watchpoint control 1 register (GTMDI_TIM_WPC1)

Register index: 11

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1			0	SSEL1			HEN1	WMC1	0	0	0	CHSEL1	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2			0	SSEL2			HEN2	WMC2	0	0	0	CHSEL2	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1749 describes the GTMDI_TIM_WPC1 register functions.

Table 1749. GTMDI_TIM_WPC1 field descriptions

Field	Description
29–28 TSS1	TIM Channel Slope Selection 1. The TSS1 field selects the slope for the TIM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
26–24 SSEL1	TIM sub-module Source Selection 1. Selects which TIM sub-module is the source for watchpoint generation. 000 TIM0 001 TIM1 010 TIM2 011 TIM3 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected TIM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TIM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TIM Filter output to generate a Watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 1749. GTMDI_TIM_WPC1 field descriptions (continued)

Field	Description
18–16 CHSEL1	TIM channel selection 1. Selects which channel within a selected TIM sub-module generates watchpoints. The selection is actually on the TIM channel input filter. 000 TIM CH0 001 TIM CH1 010 TIM CH2 011 TIM CH3 100 TIM CH4 101 TIM CH5 110 TIM CH6 111 TIM CH7
13–12 TSS2	TIM Channel Slope Selection 2. The TSS2 field selects the slope for the TIM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
10–8 SSEL2	TIM sub-module Source Selection 2. Selects which TIM sub-module is the source for watchpoint generation. 000 TIM0 001 TIM1 010 TIM2 011 TIM3 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN1 Halt Enable 2 bit controls if the selected TIM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TIM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC1 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TIM Filter output to generate a Watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
2–0 CHSEL2	TIM channel selection 2. Selects which channel within a selected TIM sub-module generates watchpoints. The selection is actually on the TIM channel input filter. 000 TIM CH0 001 TIM CH1 010 TIM CH2 011 TIM CH3 100 TIM CH4 101 TIM CH5 110 TIM CH6 111 TIM CH7

76.4.1.5 TIM watchpoint control register 2 (GTMDI_TIM_WPC2)

The GTMDI_TIM_WPC2 register shown in [Figure 1884](#) controls the Watchpoint Triggers WPT issued by TIM channels. These watchpoint triggers are used to control GTM Halt state entering, watchpoint messages and control external logic to the GTMDI. See [Figure 1927](#) for details of the logic that uses the bits in this register.

Register index: 12

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1				0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2				0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1884. TIM watchpoint control 2 register (GTMDI_TIM_WPC2)

[Table 1750](#) describes the GTMDI_TIM_WPC2 register functions.

Note: The *SHARED_WPT* and *SHARED_WPT_ST* signals mentioned in [Table 1750](#) and in other tables in this document, refer to the GTMDI top level connections. They correspond to Watchpoint trigger outputs of GTMDI and Stop/Start input signals respectively.

Table 1750. GTMDI_TIM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TIM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued. This enable refers to the output selected by the WTSEL1 field. 0 Disable Watchpoint Trigger Generation 1 Enable Watchpoint Trigger Generation

Table 1750. GTMDI_TIM_WPC2 field descriptions (continued)

Field	Description
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected TIM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TIM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TIM Channel. Other modules can select the same watchpoint trigger, in this case the result signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used to control the watchpoint trace messages for the selected TIM channel. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected TIM channel. This enable refers to the output selected by the WTSEL2 field. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TIM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.6 TOM watchpoint control register 1 (GTMDI_TOM_WPC1)

The GTMDI_TOM_WPC1 register shown in [Figure 1885](#) controls the Watchpoint Triggers issued by TOM channels. These Watchpoint triggers are used to control the GTM requests to enter Halt, issue watchpoint messages and also by logic external to the GTMDI at SoC level.

Note: *Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TOM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.*

Register index: 17

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	SSEL1			HEN1	WMC1	0	0	CHSEL1			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	SSEL2			HEN2	WMC2	0	0	CHSEL2			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1885. TOM watchpoint control 1 register (GTMDI_TOM_WPC1)

Table 1751 describes the GTMDI_TOM_WPC1 register functions.

Table 1751. GTMDI_TOM_WPC1 field descriptions

Field	Description
29–28 TSS1	TOM Channel Slope Selection 1. The TSS1 field selects the slope for the TOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
26–24 SSEL1	TOM sub-module Source Selection 1. Selects which TOM sub-module is the source for watchpoint generation. 000 TOM0 001 TOM1 010 TOM2 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected TOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 1751. GTMDI_TOM_WPC1 field descriptions (continued)

Field	Description
19–16 CHSEL1	TOM channel selection 1. Selects which channel within a selected TOM sub-module generates watchpoints. 0000 TOM CH0 0001 TOM CH1 0010 TOM CH2 0011 TOM CH3 0100 TOM CH4 0101 TOM CH5 0110 TOM CH6 0111 TOM CH7 1000 TOM CH8 1001 TOM CH9 1010 TOM CH10 1011 TOM CH11 1100 TOM CH12 1101 TOM CH13 1110 TOM CH14 1111 TOM CH15
13–12 TSS2	TOM Channel Slope Selection 2. The TSS2 field selects the slope for the TOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
10–8 SSEL2	TOM sub-module Source Selection 2. Selects which TOM sub-module is the source for watchpoint generation. 000 TOM0 001 TOM1 010 TOM2 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN1 Halt Enable 2 bit controls if the selected TOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM

Table 1751. GTMDI_TOM_WPC1 field descriptions (continued)

Field	Description
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start and Stop signals inputs to the GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
3–0 CHSEL2	TOM channel selection 2. The CHSEL2[2:0] field selects a channel within the TOM selected sub-module to generate watchpoints. 0000 TOM CH0 0001 TOM CH1 0010 TOM CH2 0011 TOM CH3 0100 TOM CH4 0101 TOM CH5 0110 TOM CH6 0111 TOM CH7 1000 TOM CH8 1001 TOM CH9 1010 TOM CH10 1011 TOM CH11 1100 TOM CH12 1101 TOM CH13 1110 TOM CH14 1111 TOM CH15

76.4.1.7 TOM watchpoint control 2 register (GTMDI_TOM_WPC2)

The GTMDI_TOM_WPC2 register shown in [Figure 1886](#) controls the Watchpoint Triggers issued by TOM channels. These watchpoints are used to force GTM into Halt state, generate watchpoint messages and control logic external to the GTMDI.

Register index: 18

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0	WTSEL1				0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																	
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0	WTSEL2				0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																	
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 1886. TOM watchpoint control 2 register (GTMDI_TOM_WPC2)

[Table 1752](#) describes the GTMDI_TOM_WPC2 register functions.

Table 1752. GTMDI_TOM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TOM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued by the TOM Channel output selected by WTSEL1. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected TOM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available Watchpoint trigger outputs which one is connected to the TOM selected channel. Other modules can select the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 1752. GTMDI_TOM_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected TOM channel. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs of the GTMDI to control Watchpoint trace messages generated by the TOM selected channel. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.8 ATOM watchpoint control 1 register (GTMDI_ATOM_WPC1)

The GTMDI_ATOM_WPC1 register shown in [Figure 1887](#) controls the Watchpoint Triggers issued by ATOM channels. These watchpoint triggers are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_ATOM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 23

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1			SSEL1				HEN1	WMC1	0	0	0	CHSEL1	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2			SSEL2				HEN2	WMC2	0	0	0	CHSEL2	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1887. ATOM watchpoint control 1 register (GTMDI_ATOM_WPC1)

[Table 1753](#) describes the GTMDI_ATOM_WPC1 register functions.

Table 1753. GTMDI_ATOM_WPC1 field descriptions

Field	Description
29–28 TSS1	ATOM Channel Slope Selection 1. The TSS1 field selects the slope for the ATOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
27–24 SSEL1	ATOM sub-module Source Selection 1. Selects which ATOM sub-module is the source for watchpoint generation. 0000 ATOM0 0001 ATOM1 0010 ATOM2 0011 ATOM3 0100 ATOM4 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected ATOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected ATOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected ATOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
18–16 CHSEL1	ATOM channel selection 1. Selects which channel in an ATOM module generates watchpoints. 000 ATOM CH0 001 ATOM CH1 010 ATOM CH2 011 ATOM CH3 100 ATOM CH4 101 ATOM CH5 110 ATOM CH6 111 ATOM CH7
13–12 TSS2	ATOM Channel Slope Selection 2. The TSS2 field selects the slope for the ATOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved

Table 1753. GTMDI_ATOM_WPC1 field descriptions (continued)

Field	Description
11–8 SSEL2	ATOM sub-module Source Selection 2. Selects which ATOM module is the source for watchpoint generation. 0000 ATOM0 0001 ATOM1 0010 ATOM2 0011 ATOM3 0100 ATOM4 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the ATOM selected channel is enabled to HALT the GTM module. If this bit is set and an event occurs on the ATOM selected channel, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected ATOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
2–0 CHSEL2	ATOM channel selection 2. The CHSEL2[2:0] field selects a channel within the TOM selected sub-module to generate watchpoints. 000 ATOM CH0 001 ATOM CH1 010 ATOM CH2 011 ATOM CH3 100 ATOM CH4 101 ATOM CH5 110 ATOM CH6 111 ATOM CH7

76.4.1.9 ATOM watchpoint control 2 register (GTMDI_ATOM_WPC2)

The GTMDI_ATOM_WPC2 register shown in [Figure 1887](#) controls the Watchpoint Triggers issued by ATOM channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 24

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1				0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2				0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1888. ATOM watchpoint control 2 register (GTMDI_ATOM_WPC2)

Table 1754 describes the GTMDI_ATOM_WPC2 register functions.

Table 1754. GTMDI_ATOM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected ATOM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued by the ATOM Channel output selected by WTSEL1. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected ATOM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_ATOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 1754. GTMDI_ATOM_WPC2 field descriptions (continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available Watchpoint trigger outputs which one is connected to the ATOM selected channel. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected ATOM channel. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs of the GTMDI to control Watchpoint trace messages generated by the ATOM selected channel. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_ATOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.10 SPEA watchpoint control 1 register 1 (GTMDI_SPEA_WPC1)

The GTMDI_SPEA_WPC1 register shown in [Figure 1889](#) controls the Watchpoint Triggers issued by SPEA channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and to control logic outside of the GTMDI module, such as SPU interface.

Note: *Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_SEP0_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.*

Register index: 29

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SSEL			0	WMC1	TSS1		0	0	HEN1	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	WMC2	TSS2		0	0	HEN2	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1889. SPEA watchpoint control 1 register (GTMDI_SPEA_WPC1)

Table 1755 describes the GTMDI_SPEA_WPC1 register functions.

Table 1755. GTMDI_SPEA_WPC1 field descriptions

Field	Description
26–24 SSEL	SPEA sub-module Source Selection. Selects which SPE sub-module is the source for watchpoint generation. 000 SPE0 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEA NIPD to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21–20 TSS1	SPEA NIPD Slope Selection. The TSS1 field selects the slope for the NIPD signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	Halt Enable 1. The HEN1 Halt Enable 1bit controls if the SPEA NIPD active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEA DIR output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 1755. GTMDI_SPEA_WPC1 field descriptions (continued)

Field	Description
5–4 TSS2	SPEA DIR Slope Selection. The TSS2 field selects the slope for the DIR signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
1 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the SPEA DIR active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

76.4.1.11 SPEA watchpoint control 2 register (GTMDI_SPEA_WPC2)

The GTMDI_SPEA_WPC2 register shown in [Figure 1890](#) controls the Watchpoint Triggers issued by SPEA. These watchpoints are used for GTM to enter Halt state, generate watchpoint trace messages and to control logic outside the GTMDI module, such as SPU interface.

Register index: 30

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1				0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2				0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1890. SPEA watchpoint control 2 register (GTMDI_SPEA_WPC2)

[Table 1756](#) describes the GTMDI_SPEA_WPC2 register functions.

Table 1756. GTMDI_SPEA_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available watchpoint triggers which one is connected to the NIPD SPEA signal. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the SPEA NIPD selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the SPEA NIPD selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_SPEA_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DIR SPEA signal. Other modules can select the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 1756. GTMDI_SPEA_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the SPEA DIR selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the SPEA DIR selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_SPEA_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.12 SPEB watchpoint control 1 register 1 (GTMDI_SPEB_WPC1)

The GTMDI_SPEB_WPC1 register shown in [Figure 1891](#) controls the Watchpoint Triggers issued by SPEB channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and to control logic outside of the GTMDI module, such as SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_SPEB_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 35

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SSEL			0	WMC1	TSS1		0	0	HEN	0
W															1	
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	WMC2	TSS2		0	0	HEN	0
W															2	
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1891. SPEB watchpoint control 1 register (GTMDI_SPEB_WPC1)

[Table 1757](#) describes the GTMDI_SPEB_WPC1 register functions.

Table 1757. GTMDI_SPEB_WPC1 field descriptions

Field	Description
26–24 SSEL	SPEB sub-module Source Selection ⁽¹⁾ . Selects which SPE sub-module is the source for watchpoint generation. 000 SPE1 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEB NIPD to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21–20 TSS1	SPEB NIPD Slope Selection. The TSS1 field selects the slope for the NIPD signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	Halt Enable 1. The HEN1 Halt Enable 1bit controls if the SPEB NIPD active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEB DIR output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
5–4 TSS2	SPEB DIR Slope Selection. The TSS2 field selects the slope for the DIR signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
1 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the SPEB DIR active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

1. The SSEL field for SPEB starts from SPE1 selection, differently from the equivalent field for SPEA which selects SPE0 instead.

76.4.1.13 SPEB watchpoint control 2 register (GTMDI_SPEB_WPC2)

The GTMDI_SPEB_WPC2 register shown in [Figure 1892](#) controls the Watchpoint Triggers issued by SPEB. These watchpoints are used for GTM to enter Halt state, generate

watchpoint trace messages and to control logic outside the GTMDI module, such as SPU interface.

Register index: 36

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1				0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2				0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1892. SPEB watchpoint control 2 register (GTMDI_SPEB_WPC2)

[Table 1758](#) describes the GTMDI_SPEB_WPC2 register functions.

Table 1758. GTMDI_SPEB_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available watchpoint triggers which one is connected to the NIPD SPEB signal. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the SPEB NIPD selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the SPEB NIPD selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_SPEB_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 1758. GTMDI_SPEB_WPC2 field descriptions (continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DIR SPEB signal. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the SPEB DIR selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the SPEB DIR selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_SPEB_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.14 DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1)

The GTMDI_DPLL_WPC1 register shown in [Figure 1893](#) controls various trace and debug features of the DPLL modules as described below.

Note: *Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_DPLL_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.*

Figure 1893. DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1)**Register index: 41**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0	0	0	0	0	0	TSEL1		0	WM	TSS1		0	0	HEN1		0
W										C1							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0	0	TSEL2B		0	0	TSEL2A		0	WM	0	0	0	0	HEN2		0
W										C2							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[Table 1759](#) describes the GTMDI_DPLL_WPC1 register functions.

Table 1759. GTMDI_DPLL_WPC1 field descriptions

Field	Description
25–24 TSEL1	Watchpoint Selection 1. The TSEL1[1:0] field selects if a watchpoint trigger is issued based on TASI or SASI events. 00 Watchpoint on TASI event 01 Watchpoint on SASI event 10 Reserved 11 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on DPLL SASI or TASI to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21–20 TSS1	TASI/SASI Slope Selection. The TSS1 field selects the slope for TASI or SASI depending on TSEL1 selection. The selected slope is monitored by watchpoints. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	The HEN1 Halt Enable 1 bit controls if the DPLL SASI or TASI selected signal is enabled to HALT the GTM module. If this bit is set and an event occurs on the DPLL selected signal, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
13–12 TSEL2B	Watchpoint Selection 2B. The TSEL2B field selects the RAM access type to match on 00 Watchpoint is issued on any access (read or write) 01 Watchpoint is issued based on write accesses 10 Watchpoint is issued based on read accesses 11 Reserved
9–8 TSEL2A	Watchpoint Selection 2A. The TSEL2A field selects if a watchpoint is issued based on the RAM1a, RAM1bc or RAM2 access. 00 Watchpoint is issued based on RAM1a access 01 Watchpoint is issued based on RAM1b access 10 Watchpoint is issued based on RAM2 access 11 Reserved

Table 1759. GTMDI_DPLL_WPC1 field descriptions (continued)

Field	Description
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables selected accesses on DPLL RAMs to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
1 HEN2	Halt Enable 2. The HEN Halt Enable bit controls if watchpoint from TSEL2 selection is enabled to HALT the GTM module. If this bit is set and an event occurs on an DPLL selected RAM, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM

76.4.1.15 DPLL watchpoint control 2 register (GTMDI_DPLL_WPC2)

The GTMDI_DPLL_WPC2 register shown in [Figure 1894](#) controls the Watchpoint Triggers issued by DPLL. These triggers are used for GTM to enter Halt state and by GTMDI external logic at SoC level. This register also controls the Watchpoint Trace messages WPM.

Register index: 42

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1894. DPLL watchpoint control 2 register (GTMDI_DPLL_WPC2)

[Table 1760](#) describes the GTMDI_DPLL_WPC2 register functions.

Table 1760. GTMDI_DPLL_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the DPLL SASI or TASI signal. Other modules can select this same WTSEL1 value, in this case the result trigger signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages related to DPLL SASI or TASI signals. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the DPLL TASI or SASI selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the DPLL TASI or SASI selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_DPLL_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DPLL RAM access signal. Other modules can select this same WTSEL2 value, in this case the result trigger signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used by the watchpoint trace messages related to DPLL RAM access events for RAM1A, RAM1B and RAM2. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 1760. GTMDI_DPLL_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the DPLL RAM selected access. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the DPLL RAM selected access event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_DPLL_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.16 DPLL watchpoint control 3 register (GTMDI_DPLL_WPC3)

The GTMDI_DPLL_WPC3 register shown in [Figure 1895](#) defines the data value observed in the selected DPLL RAM access. A match to this value causes a Watchpoint.

Register index: 43

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DMASK	0	0	0	0	RAM_DATA[23:16]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RAM_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1895. DPLL watchpoint control 3 register (GTMDI_DPLL_WPC3)

[Table 1761](#) describes the GTMDI_DPLL_WPC3 register functions.

Table 1761. GTMDI_DPLL_WPC3 field descriptions

Field	Description
28 DMASK	Data Mask. The DMASK field is a mask for the data comparator. If set, the data read or written to the memory should not be considered for watchpoint generation. 0 RAM_DATA should be compared with data from RAM access 1 RAM_DATA is masked and should not be compared
23–0 RAM_DATA	Data value. The RAM_DATA[23:0] field defines the data value that is compared against the data accessed from the selected DPLL RAM.

76.4.1.17 DPLL watchpoint control 4 register (GTMDI_DPLL_WPC4)

The GTMDI_DPLL_WPC4 register shown in [Figure 1896](#) defines the RAM address mask for the DPLL selected RAM address comparison.

Register index: 44

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RAM_ADDR_MASK											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1896. DPLL watchpoint control 4 register (GTMDI_DPLL_WPC4)

[Table 1762](#) describes the GTMDI_DPLL_WPC4 register functions.

Table 1762. GTMDI_DPLL_WPC4 field descriptions

Field	Description
11–0 RAM_ADDR_MASK	RAM Address mask. The RAM_ADDR_MASK[11:0] field is a mask applied bit by bit to the RAM_ADDR defined in the GTMDI_DPLL_WPC5 register. If mask is 1, the corresponding RAM_ADDR bit is not valid for comparison, thus it always returns a match when compared to the selected RAM address.

76.4.1.18 DPLL watchpoint control 5 register (GTMDI_DPLL_WPC5)

The GTMDI_DPLL_WPC5 register shown in [Figure 1897](#) defines the address value that is monitored during DPLL selected RAM access. Once this address value matches with the DPLL RAM address a Watchpoint is issued.

Register index: 45

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RAM_ADDR											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1897. DPLL watchpoint control 5 register (GTMDI_DPLL_WPC5)

[Table 1763](#) describes the GTMDI_DPLL_WPC5 register functions.

Table 1763. GTMDI_DPLL_WPC5 field descriptions

Field	Description
11–0 RAM_ADDR	RAM Address Compare. The RAM_ADDR[11:0] field defines the address value that is compared against the address for write and read DPLL RAM accesses. Before the comparison is executed the RAM_ADDR_MASK should be considered thus masking corresponding bits. A masked bit is always considered a match. <i>Note: DPLL module has three RAM modules with different address widths. The address value in the RAM_ADDR field and the RAM address should be right aligned in order to be compared.</i>

76.4.1.19 DPLL data trace control register (GTMDI_DPLL_DTC)

The GTMDI_DPLL_DTC register shown in [Figure 1898](#) controls data trace message transmission.

Register index: 46

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWC		0	0	RAMSEL		0	DMC	STSEL		0	0	0	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1898. DPLL data trace control register (GTMDI_DPLL_DTC)

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the DMC control bit thus controlling the Data Trace messages behavior. In case of JTAG modifies the DMC bit and Start/Stop signals occur at the same time also modifying the same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit from the GTMDI_DPLL_DTC register need to be set in order to allow external Start/Stop signals to control the DMC bit.

[Table 1764](#) describes the GTMDI_DPLL_DTC register functions.

Table 1764. GTMDI_DPLL_DTC field descriptions

Field	Description
13–12 RWC	Read/Write Control. This bit controls DPLL data trace message transmission. 00 Transmits only memory reads 01 Transmits only memory writes 10 Transmits all memory accesses 11 Reserved
9–8 RAMSEL	RAM Module Selection. This field selects a DPLL memory module for data tracing. 00 RAM1a selected for data tracing 01 RAM1b selected for data tracing 10 RAM2 selected for data tracing 11 Reserved

Table 1764. GTMDI_DPLL_DTC field descriptions (continued)

Field	Description
6 DMC	Data Trace Message Control. This bit is written by JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
5–4 STSEL	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
0 SEN	Start/Stop Enable. The SEN bit field enables the DPLL data trace message generator to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DMC Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.20 ARU watchpoint control 1 register (GTMDI_ARU_WPC1)

The GTMDI_ARU_WPC1 register shown in [Figure 1899](#) controls the Watchpoint Triggers issued by ARU Debugging Channels. These watchpoint triggers are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_ARU_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 50

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	TSS1	0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	TSS2	0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1899. ARU watchpoint control 1 register (GTMDI_ARU_WPC1)

[Table 1765](#) describes the GTMDI_ARU_WPC1 register functions.

Table 1765. GTMDI_ARU_WPC1 field descriptions

Field	Description
28 TSS1	ARU Debugging Channel 0 Data Activity Selection 1. The TSS1 bit selects the type of data activity to monitor on ARU Debugging Channel 0. 0 Any Data (do not compare data). Watchpoint generation is based on a valid data indication on ARU Debugging Channel 0 1 Compare data with expected value
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the ARU Debugging Channel 0 is enabled to HALT the GTM module. If this bit is set and an event occurs on the ARU Debugging Channel 0, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches on the ARU Debugging Channel 0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
12 TSS2	ARU Debugging Channel 1 Data Activity Selection. The TSS2 bit selects the type of data activity to monitor on the ARU Debugging Channel 1. 0 Any Data (do not compare data). Watchpoint generation is based on a valid data indication on ARU Debugging Channel Monitor 1 1 Compare data with expected value
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the ARU debugging Channel 1 is enabled to HALT the GTM module. If this bit is set and an event occurs on the ARU Debugging Channel 1, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match on the ARU Debugging Channel 1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

76.4.1.21 ARU watchpoint control 2 register (GTMDI_ARU_WPC2)

The GTMDI_ARU_WPC2 register shown in [Figure 1899](#) controls the Watchpoint Triggers issued by ARU Debugging Channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 51

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1				0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2				0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1900. ARU watchpoint control 2 register (GTMDI_ARU_WPC2)

Table 1766 describes the GTMDI_ARU_WPC2 register functions.

Table 1766. GTMDI_ARU_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the ARU Debugging Channel 0. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages for ARU Debugging Channel 0. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the ARU debugging Channel 0. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the ARU Debugging Channel 0 to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_ARU_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 1766. GTMDI_ARU_WPC2 field descriptions (continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the ARU Debugging Channel 1. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control for ARU Debugging Channel 1. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the ARU Debugging Channel 1 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the ARU Debugging Channel 1 to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_ARU_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.22 ARU watchpoint DATA0H register (GTMDI_ARU_DATA0H)

The ARU Watchpoint DATA0H high word shown in [Figure 1901](#) is used for comparison against the high word of the ARU Debugging Channel 0. The address of the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS0 register.

Register index: 52

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA0H[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA0H[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1901. ARU watchpoint DATA0H register (GTMDI_ARU_DATA0H)

[Table 1767](#) describes the GTMDI_ARU_DATA0H register functions.

Table 1767. GTMDI_ARU_DATA0H field descriptions

Field	Description
28–0 DATA0H	Watchpoint DATA0H. This field is compared against the high word of the ARU Debugging Channel 0.

76.4.1.23 ARU watchpoint DATA0L register (GTMDI_ARU_DATA0L)

The ARU Watchpoint DATA0H low word shown in [Figure 1902](#) is used for comparison against the low word of the ARU Debugging Channel 0. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS0 register.

Register index: 53

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA0L[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA0L[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1902. ARU watchpoint DATA0L register (GTMDI_ARU_DATA0L)

[Table 1768](#) describes the GTMDI_ARU_DATA0L register functions.

Table 1768. GTMDI_ARU_DATA0L field descriptions

Field	Description
28–0 DATA0L	Watchpoint DATA0L. This field is compared against the low word of the ARU Debugging Channel 0.

76.4.1.24 ARU watchpoint DATA1H register (GTMDI_ARU_DATA1H)

The ARU Watchpoint DATA1H high word shown in [Figure 1903](#), is used for comparison against the high word of the ARU Debugging Channel 1. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS1 register.

Register index: 54

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA1H[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA1H[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1903. ARU watchpoint DATA1H register (GTMDI_ARU_DATA1H)

[Table 1769](#) describes the GTMDI_ARU_DATA1H register functions.

Table 1769. GTMDI_ARU_DATA1H field descriptions

Field	Description
28–0 DATA1H	Watchpoint DATA1H. This field is compared against the high word of the ARU Debugging Channel 1.

76.4.1.25 ARU watchpoint DATA1L register (GTMDI_ARU_DATA1L)

The ARU Watchpoint DATA1L low word shown in [Figure 1904](#) is used to be compared against the low word of the ARU Debugging Channel 1. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS1 register.

Register index: 55

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA1L[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA1L[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1904. ARU Watchpoint DATA1L (GTMDI_ARU_DATA1L)

[Table 1770](#) describes the GTMDI_ARU_DATA1L register functions.

Table 1770. GTMDI_ARU_DATA1L field descriptions

Field	Description
28–0 DATA1L	Watchpoint DATA1L. This field is compared against the low word of the ARU Debugging Channel 1.

76.4.1.26 ARU data trace control register (GTMDI_ARU_DTC)

The GTMDI_ARU_DTC register shown in [Figure 1905](#) controls the ARU data trace message transmission. The GTM Module selects two ARU addresses to be monitored, using the ARU Debugging Channels 0 and 1. The data trace for these two channels is controlled by DMC1/2 bits in this register. Once enabled, messages are generated as soon as a valid data is identified in one of the ARU Debugging Channels, 0 or 1.

Note: *Start/Stop signals inputs to the GTMDI can overwrite the state of the DMC1 and DMC2 control bits thus controlling the Data Trace messages behavior. In case of JTAG modifies the DMC1 or DMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits need to be set in order to allow external Start/Stop signals to control the DMC1/2 bits.*

Register index: 56

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	STSEL1		0	0	DMC1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	STSEL2		0	0	DMC2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1905. ARU data trace control register (GTMDI_ARU_DTC)

Table 1771 describes the GTMDI_ARU_DTC register functions.

Table 1771. GTMDI_ARU_DTC field descriptions

Field	Description
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the ARU Debugging Channel 0 data trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 DMC1	Data Trace Message Control. This bit controls ARU data trace messages transmission for ARU Debugging Channel 0. This bit is written by the JTAG interface but can also be controlled at SoC level by dedicated Start/Stop signals, inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
16 SEN1	Start/Stop Enable 1. The SEN 1 bit field enables the ARU data trace message generator for ARU Debugging Channel 0 to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DMC1 Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used by the ARU Debugging Channel 1 data trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 1771. GTMDI_ARU_DTC field descriptions (continued)

Field	Description
1 DMC2	Data Trace Message Control. This bit controls ARU data trace message transmission for ARU Debugging Channel 1. This bit is written by the JTAG interface but can also be controlled at SoC level by dedicated Start/Stop signals, inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
0 SEN2	The SEN2 bit field enables the ARU data trace message generator for ARU Debugging Channel 1 to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to DMC2 Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.27 MCSA/B development control register (GTMDI_MCSA_DC and GTMDI_MCSB_DC)

The GTMDI_MCSA_DC and GTMDI_MCSB_DC registers shown in [Figure 1906](#) control various trace and debug features.

Note: Since MCSA and MCSB have a common sets of registers they are described together to avoiding duplication of similar information.

Register index: MCSA–60 / MCSB–77

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	MCSA/B_SEL				0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1906. MCSA/B development control register (GTMDI_MCSA_DC and GTMDI_MCSB_DC)

[Table 1772](#) describes the GTMDI_MCSA_DC and GTMDI_MCSB_DC register functions.

Table 1772. GTMDI_MCSA_DC and GTMDI_MCSB_DC field descriptions

Field	Description
11–8 MCSA/B_SEL	MCSA/B Selection field. The MCSA/B_SEL[3:0] selection field selects which one of the MCS cores in the GTM module are assigned as MCSA/B. 0000 MCS0 0001 MCS1 0010 MCS2 0011 MCS3 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

76.4.1.28 MCSA/B watchpoint control register (GTMDI_MCSA_WPC and GTMDI_MCSB_WPC)

The MCSA/B Watchpoint Control Registers shown in [Figure 1907](#) are used to configure MCSA/B watchpoints. These registers can be configured for halt/watchpoint based on instruction fetch, data read or data write, with address and data masking options.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC control bit thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC bit and Start/Stop signals occur at the same time also modifying the same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit needs to be asserted in order to allow external Start/Stop signals to control the WMC bit.

The watchpoint channel selection is made through the GTMDI_MCSA_CE and GTMDI_MCSB_CE by the WPCE field.

Register index: MCSA–62 / MCSB–79

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	HACH		0	WMC	0	0	HME			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HWO		0	0	0	0	0	EOC	HEN	0	STSEL		0	0	TEN	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1907. MCSA/B watchpoint control register (GTMDI_MCSA_WPC and GTMDI_MCSB_WPC)

[Table 1773](#) describes the GTMDI_MCSA_WPC and GTMDI_MCSB_WPC register functions.

Table 1773. GTMDI_MCSA_WPC and GTMDI_MCSB_WPC field descriptions

Field	Description
25–24 HACH	Halt/Watchpoint on Active Channel. The HACH field selects which type of MCS RAM access generates a watchpoint or halt. Combining this field with HWO gives GTMDI the capability of monitoring channel activity as well as accesses to specific addresses or data. 00 Match on Instruction Fetch 01 Match on Data Read 10 Match on Data Write 11 Match on Data Read or Write
22 WMC	Watchpoint Message Control. The WMC Watchpoint Message Control bit controls the watchpoint message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Watchpoints to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
19–16 HME	Halt/Watchpoint Data Mask Enable. The HME field selects which data bytes are used for hardware halt/watchpoint generation when comparing to HWD register. This register field is only meaningful if HWO bit field is [x1], otherwise the data comparison is disabled independent of HME value. 0000 All bytes of data compared xxx1 Least significant byte of data masked out xx1x Second least significant byte of data masked out x1xx Second most significant byte of data masked out 1xxx Most significant byte of data masked out
15–14 HWO	Halt/Watchpoint Operand. The HWO field selects if address and/or data matching is done. Registers WPA and WPD are used for address and data comparison, respectively. If address is not compared WPA is not use. If data is not compared then WPD is not used. 00 Comparison Disabled for address or data x1 Compare data with HWD field 1x Compare address with HWA field
8 EOC	$\overline{\text{EVT0}}$ Control. The EOC bit controls the indication on the $\overline{\text{EVT0}}$ output. See Figure 1910 , Figure 1911 , Figure 1912 and Figure 1913 for definition about watchpoint 1 and 2. 0 Indicates Watchpoint 2 occurrence 1 Indicates Watchpoint 1 occurrence
7 HEN	Halt Enable. The HEN Halt Enable bit controls if the MCSA/B Watchpoint is enabled to HALT the GTM IP module. If this bit is set and watchpoint occurs on the MCSA/B a Halt signal is generated to the GTM IP which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
5–4 STSEL	Start/Stop input signal selection. Selects the Start/Stop input that is used to control the watchpoint messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 1773. GTMDI_MCSA_WPC and GTMDI_MCSB_WPC field descriptions (continued)

Field	Description
1 TEN	Watchpoint Trigger Enable. The TEN field controls if a external trigger is issued by the MCSA/B when a watchpoint occurs. 0 Disables Trigger Generation 1 Enables Trigger Generation
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Watchpoint message to consider the Start/Stop inputs for watchpoint message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC Watchpoint Trace Message Control bit. 0 Disables trace control from Start/Stop inputs 1 Enables trace control from Start/Stop inputs

76.4.1.29 MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC)

The GTMDI_MCSA_PTC and GTMDI_MCSB_PTC registers shown in [Figure 1908](#) allow program fetch traces to be enabled and/or disabled on the occurrence of a watchpoint. If the same watchpoint is programmed to enable and disable a trace, the occurrence of the watchpoint toggles the current value of the trace flags. If one watchpoint occurs to enable trace and a different watchpoint occurs to disable trace at the same time the current value of the trace flags also toggle.

Register index: MCSA-63 / MCSB-80

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PFTS		0	0	PFTE		0	FTC		0	0	0	0	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1908. MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC)

[Table 1774](#) describes the GTMDI_MCSA_PTC and GTMDI_MCSB_PTC register functions.

Table 1774. GTMDI_MCSA_PTC and GTMDI_MCSB_PTC field descriptions

Field	Description
13–12 PFTS	Program Fetch Trace Start. This bit field allows Program fetch trace to be enabled at a watchpoint occurrence. See Figure 1910 , Figure 1911 , Figure 1912 and Figure 1913 for definition about watchpoint 1 and 2. 00 No watchpoint selected (do not start Program Fetch Trace) 01 Program Fetch Trace enabled on MCSA/B Program Watchpoint 1 occurrence 10 Program Fetch Trace enabled on MCSA/B Program Watchpoint 2 occurrence 11 Reserved
9–8 PFTE	Program Fetch Trace End. This bit field allows program trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Program Fetch Trace disabled on MCSA/B Program Watchpoint 1 occurrence 10 Program Fetch Trace disabled on MCSA/B Program Watchpoint 2 occurrence 11 Reserved
6 FTC	Fetch Trace Control. This bit controls the Fetch Trace message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Fetch access to generate a trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI 0 Fetch Messages disabled 1 Fetch Messages enabled
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Fetch message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the FTC Fetch Trace Message Control bit. 0 Disable fetch trace control from Start/Stop inputs 1 Enable fetch trace control from Start/Stop inputs

76.4.1.30 MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC)

The GTMDI_MCSA_DTC and GTMDI_MCSB_DTC registers shown in [Figure 1909](#) control which address ranges are enabled for data trace, and if reads and/or writes are traced in that range.

Register index: MCSA–64 / MCSB–81

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RC0	RC1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWT0		0	0	RWT1		0	DTC	DTS		0	DTE		SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1909. MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC)

[Table 1775](#) describes the GTMDI_MCSA_DTC and GTMDI_MCSB_DTC register functions.

Table 1775. GTMDI_MCSA_DTC and GTMDI_MCSB_DTC field descriptions

Field	Description
17 RC0	Range Control 0. Controls the range on data trace window 0. The window 0 is defined by DTAR1 register. See Figure 1915 . 0 Trace addresses outside (exclusive) of data trace window 0 1 Trace address inside (inclusive) of data trace window 0
16 RC1	Range Control 1. Controls the range on data trace window 1. The window 1 is defined by DTAR2 register. See Figure 1916 . 0 Trace addresses outside (exclusive) of data trace window 1 1 Trace address inside (inclusive) of data trace window 1
13–12 RWT0	MCSA/B Read/Write Trace Control 0. RWT0 controls whether data trace messages are generated for MCSA/B RAM read or write accesses. 00 No MCSA/B data trace messages generated for data trace window 0 01 Enable MCSA/B data read trace for MCS window 0 10 Enable MCSA/B data write trace for MCS window 0 11 Enable MCSA/B data read and write trace for data trace window 0
9–8 RWT1	MCSA/B Read/Write Trace Control 1. RWT1 controls whether data trace messages are generated for MCSA/B accesses inside MCS data trace window, and if so, whether reads, writes, or both generate the data trace messages. 00 No MCSA/B data trace messages generated for data trace window 1 01 Enable MCSA/B data read trace for MCS window 1 10 Enable MCSA/B data write trace for MCS window 1 11 Enable MCSA/B data read and write trace for data trace window 1
6 DTC	Data Trace Control. This bit controls the Data Trace message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Data access to generate a data trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI 0 Data Messages disabled 1 Data Messages enabled
5–4 DTS	Data Trace Start. This bit field allows data trace to be enabled at a watchpoint occurrence. 00 No watchpoint selected 01 Data Trace enabled on MCSA/B Watchpoint 1 occurrence 10 Data Trace enabled on MCSA/B Watchpoint 2 occurrence 11 Reserved
1–0 DTE	Data Trace End. This bit field allows data trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Data Trace disabled on MCSA/B Watchpoint 1 occurrence 10 Data Trace disabled on MCSA/B Watchpoint 2 occurrence 11 Reserved
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Data message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DTC Data Trace Message Control bit. 0 Disable data trace control from Start/Stop inputs 1 Enable data trace control from Start/Stop inputs

76.4.1.31 MCSA/B watchpoint address 1 register (GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1)

The MCSA/B Watchpoint Address Registers 1 shown in [Figure 1910](#) are used to configure the MCSA/B halt/watchpoint hardware. They are used for data and instruction fetch accesses.

Register index: MCSA–65 / MCSB–82

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HWAM													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	HWA													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1910. MCSA/B watchpoint address 1 register (GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1)

Note: The HWA field represents word addresses relative to the MCS address map, not the CPU address map. This applies to GTMDI_MCSA_WPA1, GTMDI_MCSB_WPA1, GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 registers.

[Table 1776](#) describes the GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1 register functions.

Table 1776. GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1 field descriptions

Field	Description
29–16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13–0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data addresses. The HWA field represents a word address.

76.4.1.32 MCSA/B watchpoint address 2 register (GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2)

The MCSA/B Watchpoint Address 2 Registers shown in [Figure 1911](#) are used to configure the MCSA/B halt/watchpoint hardware. They are used for data and instruction fetch accesses.

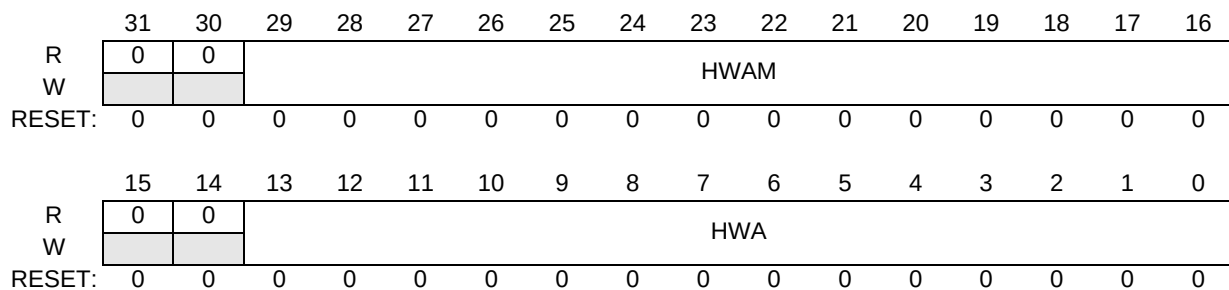
Register index: MCSA–66 / MCSB–83

Figure 1911. MCSA/B watchpoint address 2 register (GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2)

[Table 1777](#) describes the GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 register functions.

Table 1777. GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 field descriptions

Field	Description
29–16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13–0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data addresses. The HWA field represents a word address.

76.4.1.33 MCSA/B watchpoint data 1 register (GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1)

The MCSA/B Halt/Watchpoint Data 1 Registers shown in [Figure 1912](#) are used to configure the MCSA/B halt/watchpoint hardware. They are used for data accesses selection only.

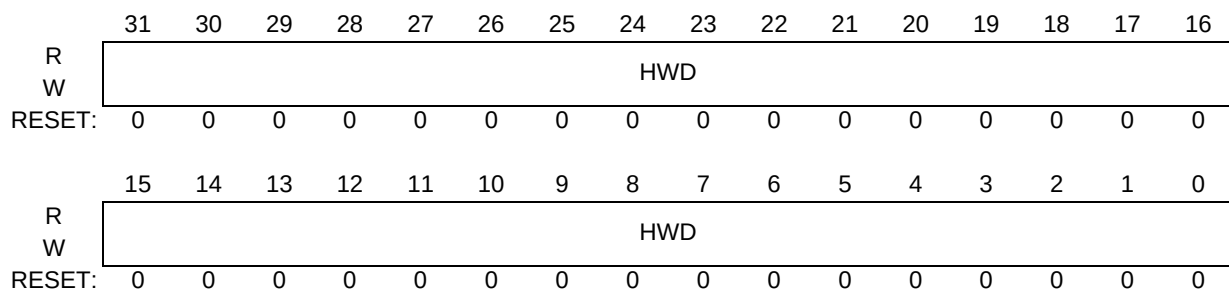
Register index: MCSA–67 / MCSB–84

Figure 1912. MCSA/B watchpoint data 1 register (GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1)

[Table 1778](#) describes the GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1 register functions.

Table 1778. GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1 field descriptions

Field	Description
31–0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the RAM data. If the operation size is 8-bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24-bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32-bits, all bits from HWD are compared against the RAM data.

76.4.1.34 MCSA/B watchpoint data 2 register (GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2)

The MCSA/B Watchpoint Data 2 Registers shown in [Figure 1913](#) are used to configure the MCSA/B halt/watchpoint hardware. They are used for data accesses selection only.

Register index: MCSA–68 / MCSB–85

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1913. MCSA/B watchpoint data 2 register (GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2)

[Table 1779](#) describes the GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2 register functions.

Table 1779. GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2 field descriptions

Field	Description
31–0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the RAM data. If the operation size is 8-bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24-bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32-bits, all bits from HWD are compared against the RAM data.

76.4.1.35 MCSA/B channel enable register (GTMDI_MCSA_CE and GTMDI_MCSB_CE)

The GTMDI_MCSA_CE and GTMDI_MCSB_CE registers shown in [Figure 1914](#) enable the monitoring for 8 MCSA/B channels independently.

Register index: MCSA–69 / MCSB–86

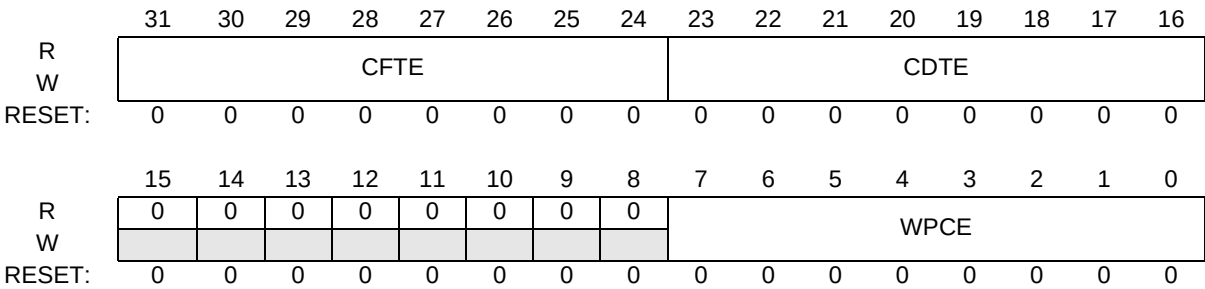


Figure 1914. MCSA/B channel enable register (GTMDI_MCSA_CE and GTMDI_MCSB_CE)

Table 1780 describes the GTMDI_MCSA_CE and GTMDI_MCSB_CE register functions.

Table 1780. GTMDI_MCSA_CE and GTMDI_MCSB_CE field descriptions

Field	Description
7–0 CFTE	MCSA/B Channel Fetch Trace Enable. The CFTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for fetch traces. If enabled, the corresponding channel Fetches are monitored by Fetch Trace messages. Start and Stop message controls are controlled by the FTC bit in the GTMDI_MCSA/B_PTC registers (Section 76.4.1.29, MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC)). 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111All Channels selected

Table 1780. GTMDI_MCSA_CE and GTMDI_MCSB_CE field descriptions (continued)

Field	Description
7–0 CDTE	MCSA/B Channel Data Trace Enable. The CDTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for data trace messages. If enabled, the corresponding channel Data Accesses are monitored by Data Trace messages. Start and Stop message are controlled by the DTC bit in the GTMDI_MCSA/B_DTC registers (Section 76.4.1.30, MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC)) and by the Data trace window. 00000000 No channel selected for trace xxxxxxx1 Channel [0] selected xxxxxx1x Channel [1] selected xxxxx1xx Channel [2] selected xxxx1xxx Channel [3] selected xxx1xxxx Channel [4] selected xx1xxxxx Channel [5] selected x1xxxxxx Channel [6] selected 1xxxxxxx Channel [7] selected 11111111 All Channels selected
7–0 WPCE	MCSA/B Channel Enable. The WPCE[7:0] is a one hot bit field which defines the channel or channels to be monitored for watchpoint, data and fetch traces. The messages controlled by this bit must be also enabled and disabled according to occurrence of Watchpoints and Address Ranges. 00000000 No channel selected for trace xxxxxxx1 Channel [0] selected xxxxxx1x Channel [1] selected xxxxx1xx Channel [2] selected xxxx1xxx Channel [3] selected xxx1xxxx Channel [4] selected xx1xxxxx Channel [5] selected x1xxxxxx Channel [6] selected 1xxxxxxx Channel [7] selected 11111111 All Channels selected

76.4.1.36 MCSA/B data trace address range 1 register (GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1)

The GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 registers shown in [Figure 1915](#) define a data trace address range. The address range is used by the MCSA/B and enables the range for reads and/or writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 0.

Figure 1915. Data trace address range 1 register (GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1)

Register index: MCSA-71 / MCSB-88

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: The DTSA and DTEA fields represent word addresses relative to the MCS address map, not the CPU address.

[Table 1781](#) describes the GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 register functions.

Table 1781. GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 field descriptions

Field	Description
29-18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSA/B data trace window.
13-2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSA/B data trace window.

76.4.1.37 MCSA/B data trace address range 2 register (GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2)

The GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2 registers shown in [Figure 1916](#) define a data trace address range. The address range is used by the MCSA/B and enables the range for reads and/or writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 1.

Register index: MCSA-72 / MCSB-89

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1916. MCSA data trace address range 2 register (GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2)

Note: DTSA and DTEA fields represent word addresses which are included in the address range.

[Table 1782](#) describes the GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR12 register functions.

Table 1782. GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2 field descriptions

Field	Description
29–18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSA/B data trace window.
13–2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSA/B data trace window.

76.4.1.38 TBU0 watchpoint control 1 register (GTMDI_TBU0_WPC1)

The GTMDI_TBU0_WPC1 register shown in [Figure 1917](#) controls the Watchpoint issued by TBU0 GTM Time Base Units (TBU). These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU0_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 94

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	TSS1				0	0	0	0	HEN1	WMC1	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TSS2				0	0	0	0	HEN2	WMC2	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1917. TBU0 watchpoint control 1 register (GTMDI_TBU0_WPC1)

[Table 1783](#) describes the GTMDI_TBU0_WPC1 register functions.

Table 1783. GTMDI_TBU0_WPC1 field descriptions

Field	Description
30–28 TSS1	TBU0 Watchpoint 1 Compare Selection. The TSS1[2:0] field controls the type of comparison used for TBU0 watchpoint generation. 000 TBU0 24-bit update (do not compare data) 001 TBU 27-bit update (do not compare data) 010 Compare TBU0 value in 24-bit mode with expected data⁽¹⁾ 011 Compare TBU0 value in 27-bit mode with expected data 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU0 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU0 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
14–12 TSS2	TBU0 Watchpoint 2 Compare Selection 2. The TSS2[2:0] field controls the type of comparison used for TBU0 watchpoint generation. 000 TBU0 24-bit update (do not compare data) 001 TBU 27-bit update (do not compare data) 010 Compare TBU0 value in 24-bit mode with expected data¹ 011 Compare TBU0 value in 27-bit mode with expected data 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU0 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU0, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

1. Only bits from 26 to 3 from register GTMDI_TBU0_DATA are used for comparison

76.4.1.39 TBU0 watchpoint control 2 register (GTMDI_TBU0_WPC2)

The GTMDI_TBU0_WPC2 shown in [Figure 1918](#) register controls the Watchpoint Triggers issued by TBU0. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 95

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1918. TBU0 watchpoint control 2 register (GTMDI_TBU0_WPC2)

[Table 1784](#) describes the GTMDI_TBU0_WPC2 register functions.

Table 1784. GTMDI_TBU0_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU0 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the TBU0 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU0 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU0_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 1784. GTMDI_TBU0_WPC2 field descriptions (continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU0 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU0 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU0 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU0_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.40 TBU0 watchpoint DATA register (GTMDI_TBU0_DATA)

The TBU0 Watchpoint DATA shown in [Figure 1919](#) is used for watchpoints generation for matching with the TBU0 value.

Register index: 96

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	TBU0_DATA[26:16]										
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU0_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1919. TBU0 watchpoint DATA register (GTMDI_TBU0_DATA)

[Table 1785](#) describes the GTMDI_TBU0_DATA register functions.

Table 1785. GTMDI_TBU0_DATA field descriptions

Field	Description
26–0 TBU0_DATA	TBU0 Watchpoint DATA. The field TBU0_DATA is used to compare against the TBU0 data.

76.4.1.41 TBU1 watchpoint control 1 register (GTMDI_TBU1_WPC1)

The GTMDI_TBU1_WPC1 register shown in [Figure 1920](#) controls the Watchpoint issued by TBU1 GTM Time Base Units. These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU1_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 101

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1920. TBU1 watchpoint control 1 register (GTMDI_TBU1_WPC1)

[Table 1786](#) describes the GTMDI_TBU1_WPC1 register functions.

Table 1786. GTMDI_TBU1_WPC1 field descriptions

Field	Description
29–28 TSS1	TBU1 Watchpoint 1 Compare Selection. The TSS1[1:0] field controls the type of comparison used for TBU1 watchpoint generation. 00 TBU1 update (do not compare data) 01 Compare TBU1 value with expected data 10 Reserved 11 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU1 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU1 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM

Table 1786. GTMDI_TBU1_WPC1 field descriptions (continued)

Field	Description
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
13–12 TSS2	TBU1 Watchpoint 2 Compare Selection 2. The TSS2[1:0] field controls the type of comparison used for TBU1 watchpoint generation. 00 TBU1 update (do not compare data) 01 Compare TBU1 value with expected data 10 Reserved 11 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU1 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU1, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

76.4.1.42 TBU1 watchpoint control 2 register (GTMDI_TBU1_WPC2)

The GTMDI_TBU1_WPC2 register shown in [Figure 1921](#) controls the Watchpoint Triggers issued by TBU1. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 102

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				0	0	0	0	0	0			0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0	0	0	0	0	0			0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1921. TBU1 watchpoint control 2 register (GTMDI_TBU1_WPC2)

[Table 1787](#) describes the GTMDI_TBU1_WPC2 register functions.

Table 1787. GTMDI_TBU1_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU1 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the TBU1 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU1 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU1_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU1 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 1787. GTMDI_TBU1_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU1 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU1 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU1_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.43 TBU1 watchpoint DATA register (GTMDI_TBU1_DATA)

The TBU1 Watchpoint DATA shown in [Figure 1922](#) is used for watchpoints generation for matching with the TBU1 value.

Register index: 103

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TBU1_DATA[23:16]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU1_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1922. TBU1 watchpoint DATA register (GTMDI_TBU1_DATA)

[Table 1788](#) describes the GTMDI_TBU1_DATA register functions.

Table 1788. GTMDI_TBU1_DATA field descriptions

Field	Description
23–0 TBU1_DATA	TBU1 Watchpoint DATA. The field TBU1_DATA[23:0] is used to compare against the TBU1 data.

76.4.1.44 TBU2 watchpoint control 1 register (GTMDI_TBU2_WPC1)

The GTMDI_TBU2_WPC1 register shown in [Figure 1923](#) controls the Watchpoint issued by TBU2 GTM Time Base Units. These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: *Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2*

control bits from the GTMDI_TBU2_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 108

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1923. TBU2 watchpoint control 1 register (GTMDI_TBU2_WPC1)

[Table 1789](#) describes the GTMDI_TBU2_WPC1 register functions.

Table 1789. GTMDI_TBU2_WPC1 field descriptions

Field	Description
29–28 TSS1	TBU2 Watchpoint 2 Compare Selection. The TSS1[1:0] field controls the type of comparison used for TBU2 watchpoint generation. 00 TBU2 update (do not compare data) 01 Compare TBU2 value with expected data 10 Reserved 11 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU2 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU2 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU2 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
13–12 TSS2	TBU2 Watchpoint 2 Compare Selection 2. The TSS2[1:0] field controls the type of comparison used for TBU2 watchpoint generation. 00 TBU2 update (do not compare data) 01 Compare TBU2 value with expected data 10 Reserved 11 Reserved

Table 1789. GTMDI_TBU2_WPC1 field descriptions (continued)

Field	Description
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU2 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU2, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU2 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

76.4.1.45 TBU2 watchpoint control 2 register (GTMDI_TBU2_WPC2)

The GTMDI_TBU2_WPC2 register shown in [Figure 1924](#) controls the Watchpoint Triggers issued by TBU2. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 109

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1924. TBU2 watchpoint control 2 register (GTMDI_TBU2_WPC2)

[Table 1790](#) describes the GTMDI_TBU2_WPC2 register functions.

Table 1790. GTMDI_TBU2_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU2 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the TBU2 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU2 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU2_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU2 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 1790. GTMDI_TBU2_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU2 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU2 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU2_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

76.4.1.46 TBU2 watchpoint DATA register (GTMDI_TBU2_DATA)

The TBU2 Watchpoint DATA shown in [Figure 1925](#) is used for watchpoints generation for matching with the TBU2 value.

Register index: 110

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TBU2_DATA[23:16]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU2_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1925. TBU2 watchpoint DATA register (GTMDI_TBU2_DATA)

[Table 1791](#) describes the GTMDI_TBU2_DATA register functions.

Table 1791. GTMDI_TBU2_DATA field descriptions

Field	Description
23–0 TBU2_DATA	TBU2 Watchpoint DATA. The field TBU2_DATA[23:0] is used to compare against the TBU2 data.

76.4.2 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in [Table 1745](#). Unimplemented registers should not be accessed.

76.5 Functional description

This section describes in detail the main functionality of the GTMDI module.

76.5.1 Debug enable logic

In order to enable the GTM to enter Halt state, which is the module debug state, there are some registers that need to be set. The sources that can put GTM in halt state are events from TIM, TOM, ATOM, MCS and several other internal GTM modules that are selected and controlled by the GTMDI_DC register bits shown in [Figure 1881](#). The system debug request external signal is also a source of halts, as well as direct JTAG write to control register. [Figure 1926](#) describes the logic behind the Halt signal generated by GTMDI. See [Figure 1881](#) for a description of the register bits mentioned in [Figure 1926](#).

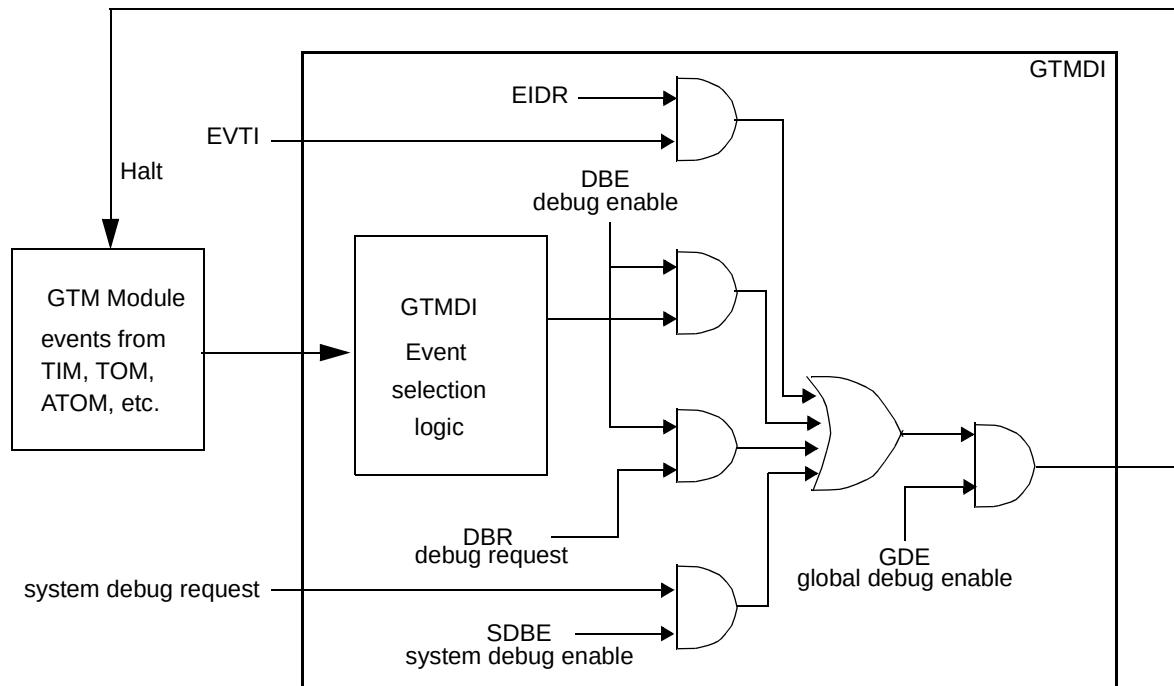


Figure 1926. GTM Halt signal enable logic

76.5.2 TIM-TOM-ATOM selection and control logic

The TIM sub-modules can be selected to generate signals that control the watchpoint messages, Halt GTM and watchpoint outputs from the GTMDI module. [Figure 1927](#) describes the TIM selection logic for generation of watchpoint messages, GTM halt and external watchpoint triggers. This logic is duplicated thus allowing independent selection of two TIM filtered inputs. [Figure 1927](#) also shows the register bits that control the selection process. Watchpoint Logic 1 and Watchpoint Logic 2 are two implementations for the same logic.

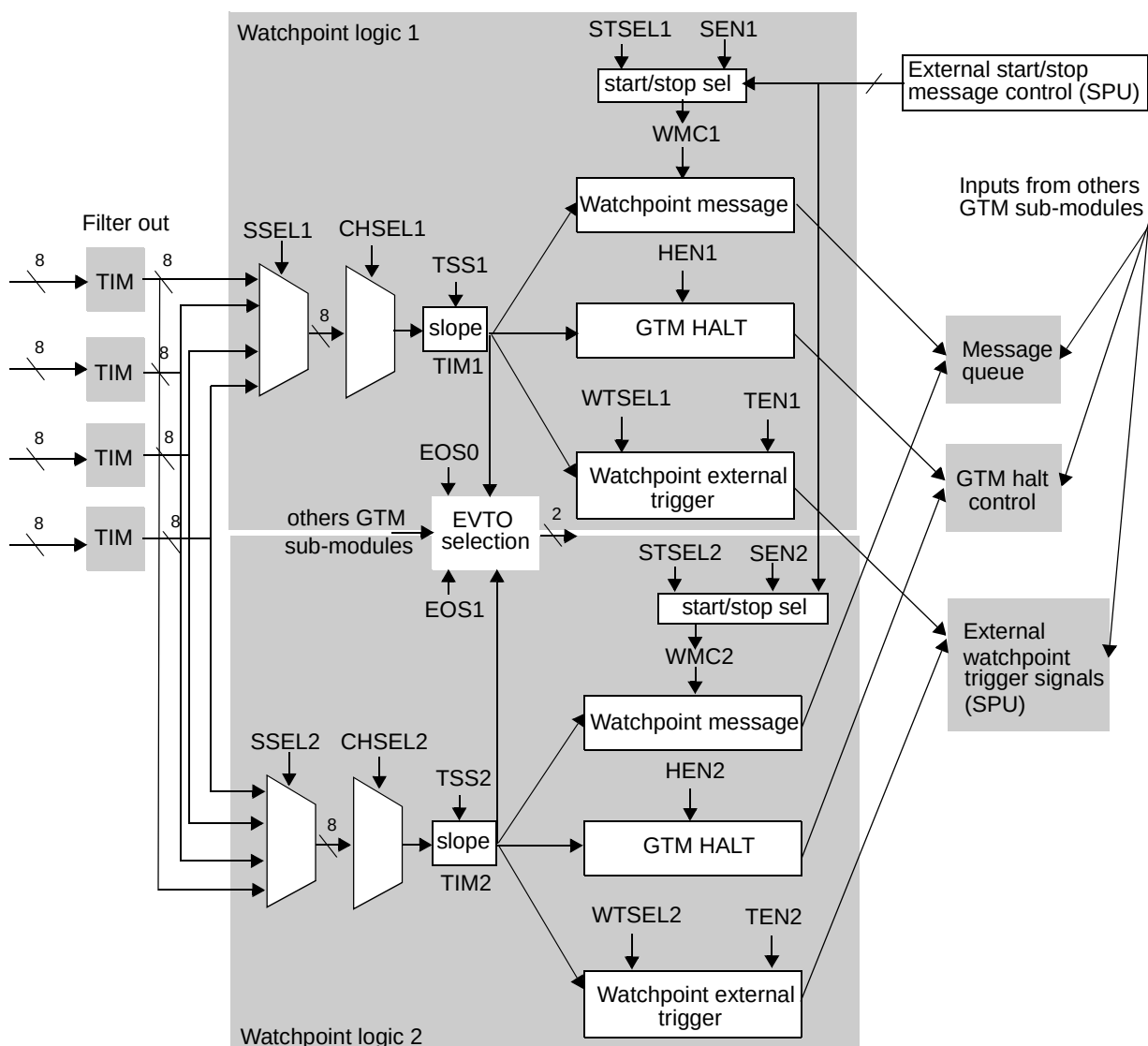


Figure 1927. TIM selection logic

Note: *Figure 1927* shows four instances of the TIM sub-module which corresponds to the GTM103 IP configuration. Different configurations may have different number of TIM sub-modules, thus SSEL1/2 signals have to be able to select different number of TIM sub-modules.

The selection logic for TOM and ATOM sub-modules is similar to the TIM logic. The register names are also kept the same for easy correlation of functionality.

76.5.3 GTMDI reset configuration

76.5.3.1 Enabling GTMDI class 1 operation

The GTMDI Class 1 features are always enabled after exiting the JTAG TEST-LOGIC-RESET state. But they are disabled while the JTAG is in this state. Thus, allowing low power mode. The registers are always reset while in the JTAG TEST-LOGIC-RESET state.

76.5.3.2 Enabling GTMDI class 3 operation

After exiting JTAG TEST-LOGIC-RESET state, the GTMDI classes 1 and 3 are enabled, otherwise the GTMDI class 3 features are disabled, entering in the Disable-Port Mode, thus no trace output is provided, and auxiliary port output pins are disabled.

76.5.4 Message Data Bus —interface with NAR module

76.5.4.1 Output messages

[Table 1792](#) describes the messages that the GTMDI can transmit on the Message Data Bus.

Table 1792. GTMDI Nexus trace message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Debug Status Message	6	6	fixed	TCODE	Value = 0
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	32	32	fixed	STATUS	Value of the Development Status register GTMDI_DS, see Table 1793
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message	6	6	fixed	TCODE	Value = 8
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	4	4	fixed	ETYPE	Always 0 for GTMDI
	15	15	fixed	ECODE	Error Code, see Table 1794
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
MCS Data Trace, Data Write with Sync Message	6	6	fixed	TCODE	Value = 56
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1=MCSB)
	3	3	fixed	CHN	MCS Channel number
	14	14	fixed	ADDR	Full address of the memory location written
	32	32	fixed	DATA	Data value written
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

Table 1792. GTMDI Nexus trace message formats (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
MCS Data Trace, Data Read with Sync Message	6	6	fixed	TCODE	Value = 57
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1=MCSB)
	3	3	fixed	CHN	MCS Channel number
	14	14	fixed	ADDR	Full address of the memory location read
	32	32	fixed	DATA	Data value read
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message ⁽¹⁾	6	6	fixed	TCODE	Value = 15
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	15	15	fixed	WPHIT	Number indicating watchpoint source (binary coded), see Table 1795 , Table 1796 and Table 1797
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Fetch Trace	6	6	fixed	TCODE	Value = 58
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1= MCSB)
	3	3	fixed	CHN	MCS Channel number
	6	6	fixed	FSCNT ⁽²⁾	Fetch Sequential Counter. Value 0 indicates there is no relationship between the last fetch trace message and the current message. The counter rolls back to one when a non sequential address is fetched. The counter increments every time a sequential address is fetched. When counter reaches 63 a message is sent and the counter rolls back to one. If an ERROR message is sent the counter rolls back to zero.
	14	14	fixed	FADDR	First element address of the next fetch sequence
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Fetch Trace End (fetch trace was turned off, no faddr field is sent)	6	6	fixed	TCODE	Value = 59
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1= MCSB)
	3	3	fixed	CHN	MCS Channel number
	6	6	fixed	FSCNT	The message is sent with the current value in the FSCNT counter and the counter rolls back to zero after the message is sent.
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

Table 1792. GTMDI Nexus trace message formats (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
ARU Data Trace	6	6	fixed	TCODE	Value = 60
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	CHN	ARU Debugging Channel number (0/1)
	58	58	fixed	DATA	Data value read {ARU_DATAH, ARU_DATA_L}
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
DPLL Data Trace, Data Read/Write	6	6	fixed	TCODE	Value = 61
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	24	24	fixed	DATA	Data value read/written
	12	12	fixed	ADDR	Full address of the memory location written
	1	1	fixed	RW	Read 1, Write 0
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Watchpoint Hit Message conforms to the IEEE-ISTO 5001-2003 standard, however, other messages conform to the IEEE-ISTO 5001-2011 standard.
2. FSCNT is the sequential instruction fetch counter that indicates the number of sequential instruction fetches executed by the MCS core for a channel. See [Section 76.5.9.2, Fetch trace state machine](#) for more information.

Table 1793. Debug status field format

STATUS bit	Selected source
12:0	HS2 field of GTMDI_DS register
14:13	Reserved
15	STP bit of GTMDI_DS register
28:16	HS1 field of GTMDI_DS register
30:29	Reserved
31	HLT bit of GTMDI_DS register

Table 1794. Error field format

ECODE bit	Selected source
2:0	TIM, TOM, ATOM
5:3	ARU Watchpoint, DPLL Watchpoint, SPE Watchpoint
7:6	ARU Data Trace, DPLL Data Trace
9:8	MCSA Watchpoint, MCSB Watchpoint
11:10	MCSA Data Trace, MCSB Data Trace

Table 1794. Error field format (continued)

ECODE bit	Selected source
13:12	MCSA Fetch Trace, MCSB Fetch Trace
14	TBU Watchpoint

Table 1795. Watchpoint field WPHIT[11:0] format for WPHIT[14:12]= 0b000

WPHIT bit	Selected source
0	ATOM 1 Negedge
1	ATOM 1 Posedge
2	ATOM 0 Negedge
3	ATOM 0 Posedge
4	TOM 1 Negedge
5	TOM 1 Posedge
6	TOM 0 Negedge
7	TOM 0 Posedge
8	TIM 1 Negedge
9	TIM 1 Posedge
10	TIM 0 Negedge
11	TIM 0 Posedge

Table 1796. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b001

WPHIT bit	Selected source
0	Reserved
1	Reserved
2	Reserved
3	DPLL (memory watchpoint)
4	DPLL (SASI)
5	DPLL (TASI)
6	ARU ch1
7	ARU ch0
8	MCSB1
9	MCSB0
10	MCSA1
11	MCSA0

Table 1797. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b010

WPHIT bit	Selected source
0	SPEA (DIR NEGEDGE)
1	SPEA (DIR POSEDGE)
2	SPEA (NIPD NEGEDGE)
3	SPEA (NIPD POSEDGE)
4	SPEB (DIR NEGEDGE)
5	SPEB (DIR POSEDGE)
6	SPEB (NIPD NEGEDGE)
7	SPEB (NIPD POSEDGE)
11:8	Reserved

Table 1798. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b011

WPHIT - bit	Selected source
0	TBU0 Watchpoint 1
1	TBU0 Watchpoint 2
2	TBU1 Watchpoint 1
3	TBU1 Watchpoint 2
4	TBU2 Watchpoint 1
5	TBU2 Watchpoint 2
11:6	Reserved

76.5.4.1.1 Temporal ordering of transmitted messages

All messages are sent out in the sequence their related event actually occurred. The GTM interface with the GTMDI provides the GTM signals to be monitored and captured. At the occurrence of any event that would cause a message, a snapshot of all information needed by the message formatter to generate a message is queued. Doing so, several events from either MCS, TIM, ATOM, ARU, etc., can be queued at the same time.

76.5.4.1.2 Interleaved messages

Two MCS cores, TIM channels and other sources can be enabled for trace. The GTMDI formats and queues all messages in the order they are generated. Since there can be message inversion among sources during the send process, if enabled by the TSFT bit in [Figure 1881](#) or by the NAR, the messages are sent along with a timestamp. This timestamp is globally synchronized among all Nexus clients at SoC level.

76.5.5 IEEE 1149.1 (JTAG) input port

The GTMDI block uses the IEEE 1149.1 TAP for accessing registers. This port is shared among all TAP controllers on the SoC. This sharing is controlled by the value in the instruction register and client select control register. The GTMDI implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1 state

machine shown in [Figure 1929](#). The Nexus controller state machine, shown in [Figure 1930](#), is defined by the IEEE-ISTO 5001-2002 standard.

The instructions implemented by the GTMDI TAP controller are listed in [Table 1799](#). If any other instruction is received it is treated as a BYPASS.

Table 1799. Implemented instructions

Instruction name	Private/ public	Opcode	Description
NEXUS-ENABLE	Public	4b0000	Activate Nexus controller state machine to read and write GTMDI registers
BYPASS	Public	4b1111	Implements a single shift register stage providing a minimum serial length between TDI and TDO

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in [Figure 1928](#). This applies for the instruction register and all Nexus tool-mapped registers.

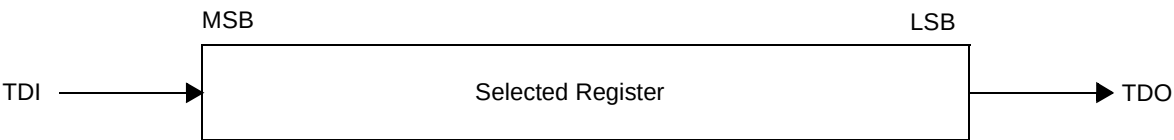
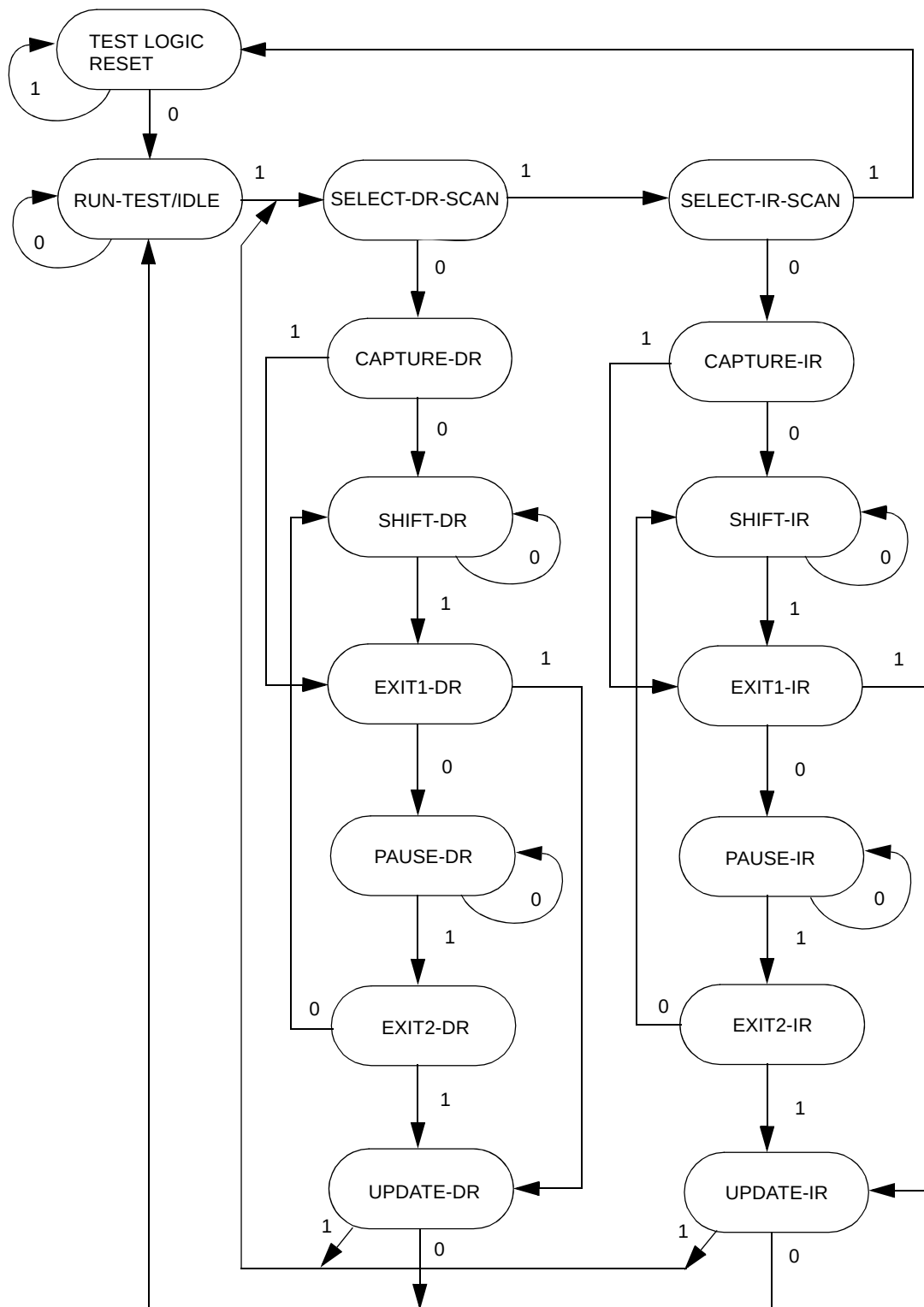


Figure 1928. Shifting data into register

76.5.5.1 Enabling the GTMDI TAP controller

Assertion of the $\overline{\text{TRST}}$ input resets all TAP controllers on the SoC. The controllers are loaded with the BYPASS instruction upon exit of the TEST-LOGIC-RESET JTAG controller state. Loading the NEXUS-ENABLE instruction grants access to Nexus debug. GTMDI acts as BYPASS by loading any other instruction. The TDO bit is driven with the value of the instruction register while in SHIFT-IR state.

Figure 1929. IEEE 1149.1 16-State finite state machine



NOTE: The value shown adjacent to each state transition in this figure represents the value of the TMS bit at the time of a rising edge of TCK.

76.5.5.2 Loading NEXUS-ENABLE instruction

Access to the GTMDI registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. The current instruction value is loaded into the IEEE 1149.1 shifter in the CAPTURE-IR state. The instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in [Figure 1930](#), transitions to the REG_SELECT state. [Table 1800](#) illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

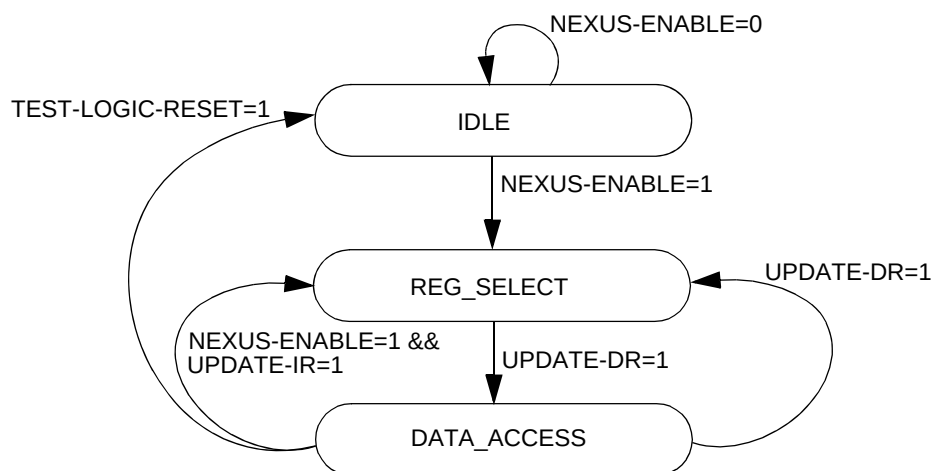


Figure 1930. Nexus controller state machine

Table 1800. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1 controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction
4 TCKs				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	NEXUS-ENABLE ready to be read

76.5.5.3 Selecting an IEEE 1149.1 register

When the NEXUS-ENABLE instruction is loaded in the GTMDI TAP controller instruction register, the input port allows development tool access to all GTMDI registers. Each register has a 7-bit address index. Some clients have registers with the same index as registers in other clients. Therefore, to access the registers of a particular client, the Client Select Control register must be set for that client.

All register access is performed via the SELECT-DR-SCAN path. The Nexus state machine defaults to the register select state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in [Figure 1931](#). The read/write control bit is set to 1 for writes and 0 for reads. The current value of the Controller Command Input is captured into the IEEE 1149.1 shifter during the CAPTURE-DR state while the Nexus controller state machine is in the REG_SELECT state.

Figure 1931. IEEE 1149.1 controller command input

MSB		LSB
7-bit register index		R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the current register value is loaded into the IEEE 1149.1 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

[Table 1801](#) illustrates a sequence which writes a 32-bit value to a register.

Table 1801. Writing to a register

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1 controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
48	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI
49	1	UPDATE-DR	DATA_ACCESS	Value written to register
50	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register

76.5.6 Nexus Class 1 development support

The GTMDI contains a number of hardware hooks that aid in the development of software for the GTM MCS cores. This features described in this section are compliant with the following Nexus Class 1 features:

- Enter a debug mode at reset negation
- Enter a debug mode during normal execution
- Set halts or watchpoints

Refer to [Section 76.6, Initialization/application information](#) for how the following functions provide these features.

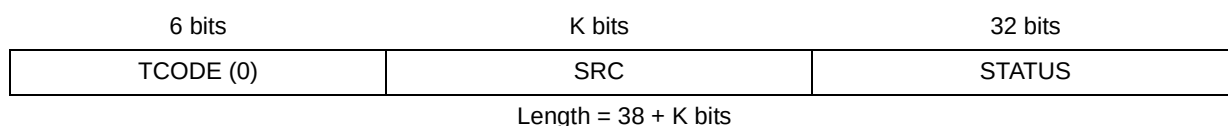
76.5.7 Debug status

The GTMDI module provides the Debug Status via the NAR SoC level modules.

76.5.7.1 Messaging

The GTMDI block provides debug status messaging using IEEE-ISTO 5001-2002 standard-defined public messages. If the message queue is full and a new message has to be sent, this message is lost and an ERROR message is queued instead, indicating the loss of trace messages. The debug status message format is shown in [Figure 1932](#).

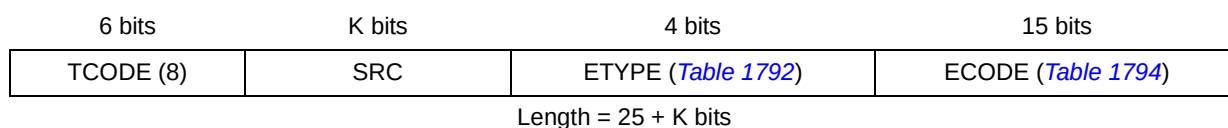
Figure 1932. Debug status message format



76.5.7.2 Error messages

A debug status overrun error event is queued if a debug status event occurs while the Event Queue is full and is not able to store snapshots. The error message is stored to the event queue indicating a message was lost. The error message is sent as soon as the client is enabled for transmission. The error message format is shown in [Figure 1933](#).

Figure 1933. Debug status error message format



Note: The timestamp value in an Error Message does not contain valid information.

76.5.8 Data trace

MCS cores, ARU and DPLL perform loads and stores to RAM memory modules. The GTMDI block gathers information about these accesses and generate trace messages if enabled.

76.5.8.1 MCS data write message

The MCS data write message contains the data write value and the address of the RAM target location. The data write message format is shown in [Figure 1934](#).

Figure 1934. Data write message format

6 bits	K bits	1 bit	3 bits	14 bits	32 bits
TCODE (56)	SRC	MCSN	CHN	ADDR	DATA

Length = 56+K bits

76.5.8.2 MCS data read message

The MCS data read message contains the data read value and the address of the RAM target location. The data read message format is shown in [Figure 1935](#).

Figure 1935. Data read message format

6 bits	K bits	1 bit	3 bits	14 bits	32 bits
TCODE (57)	SRC	MCSN	CHN	ADDR	DATA

Length = 56+K bits

76.5.8.3 MCS data trace operation

Data tracing is performed by snooping a dedicated MCS/Nexus interface for RAM read and write cycles. Data trace functions are enabled by setting the appropriate fields in the following registers:

- MCSx Development Control (GTMDI_MCSx_DC) Register
- MCSx Data Trace Control (GTMDI_MCSx_DTC) Register
- MCSx Data Trace Address Range (GTMDI_MCSx_DTAR) Register

For details on register configuration, refer to [Section 76.4.1, Register descriptions](#). The MCSx data tracing have independent Data Trace Address Range registers which allows more flexibility. An Error condition may occur in case of Queue full event. Data trace flow is depicted in [Figure 1936](#).

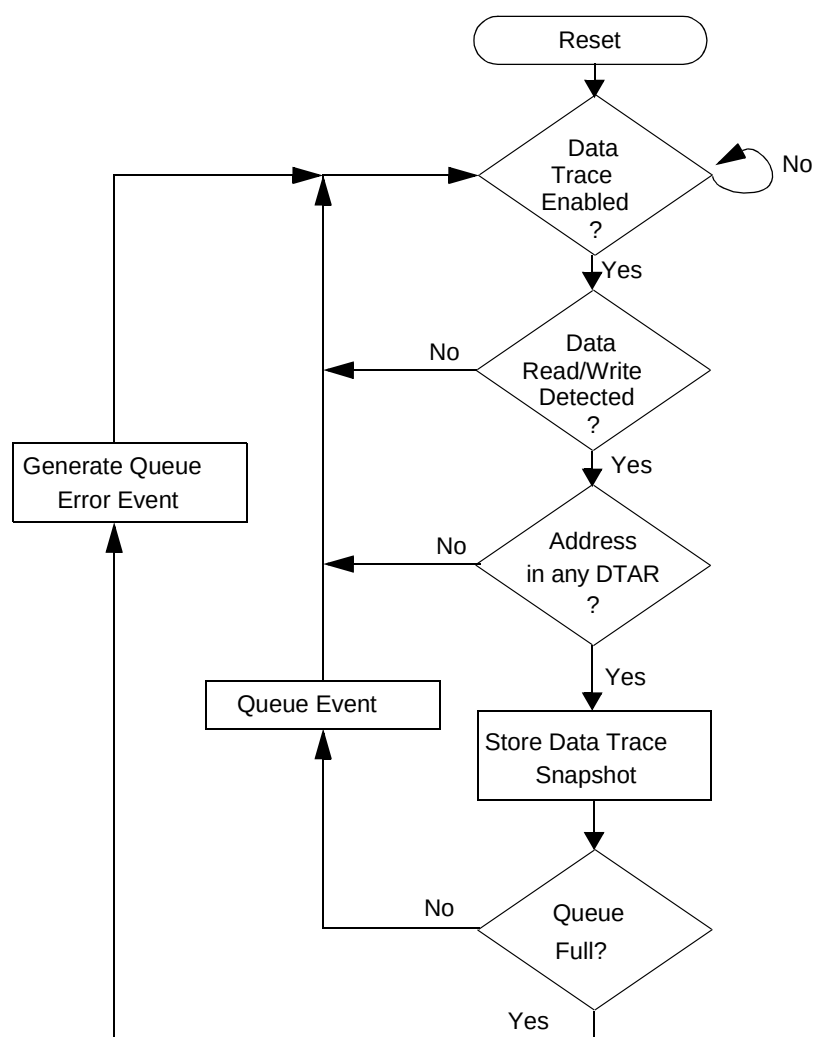


Figure 1936. GTMDI generic data trace flow diagram

76.5.8.3.1 MCS data trace windowing

Data trace windowing is provided so that the development tool can decrease the auxiliary port usage by limiting the accesses that are traced. Data trace windowing is achieved via the address range defined by the DTSA and the DTEA fields of the DTAR register. RAM accesses are traced if their address fall in any of these ranges and the specific range is enabled in the data trace control register. Data read and/or data write trace may be enabled via the Read/Write Trace Control fields in the data trace control registers.

Data trace ranges are 32-bit aligned. This alignment is done by making the two least significant bits of the DTSA and DTEA fields read only. Since the two least significant bits of DTSA and DTEA are read only and default to different values, DTSA can never be equal to DTEA. [Figure 1937](#) shows the relationship between the DTSA and DTEA fields.

Figure 1937. Data trace address range options

Programmed Value	Range Selected
DTSA ≤ DTEA	[DTSA: DTEA]
DTSA > DTEA	All addresses are out of range ⁽¹⁾

1. Since all addresses are considered to be out of range, a Data Trace Event may be recognized if the DTC register is programmed so that addresses out of range are queued.

Note: Before sending through the NAR interface, the MCS, DPLL or ARU Data Trace message is buffered. Due to bandwidth constraints in the NAR interface and in the data trace buffering, this data trace message can be lost when superseded by another data trace message from the same source. In this case an error message is sent. See [Table 1792](#) for more details about GTMDI NAR messages.

76.5.8.4 DPLL data trace

DPLL Data Trace is a straight forward implementation in which the trace is enabled using the DMC bit in the [Figure 1898](#). It is possible to select individual DPLL RAM modules such as RAM1a, RAM1b or RAM2 for tracing using the RAMSEL bit in the same register. Only one memory module is selected for tracing at a given time. Start/Stop inputs from the SPU module can be used to control the DPLL Data Trace messages as well as JTAG interface. This is achieved by controlling the DMC bit which can be done from SPU or JTAG interface. See [Table 1764](#) for more details about the bits that should be used in this process.

76.5.8.5 ARU data trace

The GTM module selects two ARU addresses to be monitored, using the ARU Debugging Channels 0 and 1, see the GTM documentation for details on how to program these channels. The data trace for these two channels is controlled by DMC1/2 bits in the GTMDI_ARU_DTC, see [Section 76.4.1.26, ARU data trace control register \(GTMDI_ARU_DTC\)](#). Once enabled, messages are generated as soon as a valid data is identified in one of the ARU Debugging Channels, 0 or 1.

76.5.9 Fetch trace

Fetch trace tries to record all instruction fetch accesses done by an MCS channel. Fetch trace is handled independently for each MCS channel, therefore what happens to a certain channel does not affect other channels since all channels run independently. Even error conditions are handled independently for each channel.

Fetch trace can be enabled/disabled by two ways:

- By watchpoint hit, if the watchpoint is configured to do so
- Directly by register

When a fetch access hits a watchpoint and the last enables fetch trace (having it disabled before the watchpoint hit), that access is ignored by the fetch trace logic.

76.5.9.1 Enabling MCS program fetch trace

Program fetch trace for the MCS is enabled on a channel by channel basis. This selective enabling allows the debug tool to control the bandwidth for the transmitted messages. The program fetch trace is enabled through a JTAG register, see [Figure 1908](#). Since the registers

may be changed in the middle of a program execution, program fetch trace may be enabled or disabled during program execution as well.

76.5.9.2 Fetch trace state machine

The fetch trace state machine is composed basically by three states: *start*, *running* and *end*. When the fetch trace is enabled, fetch state machine goes to the *start* state. [Figure 1938](#) describes the fetch trace machine state diagram.

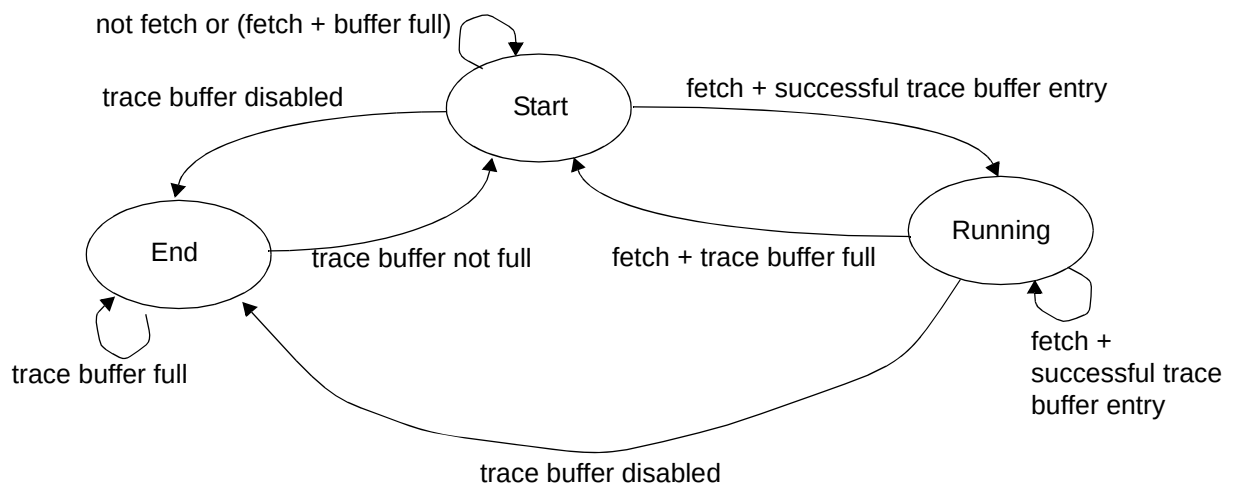


Figure 1938. Fetch trace state machine state diagram

76.5.9.2.1 Fetch trace start state

In this state, the fetch trace logic waits for a fetch access to occur. When a fetch access occurs, the fetch trace logic tries to insert (into the corresponding MCS trace buffer) a fetch trace message with FSCNT=0 and the FADDR field equal to the fetch access address. It is possible that the corresponding MCS trace message buffer is full when fetch trace logic tries to insert the message. In this case, the fetch trace logic ignores the current fetch access, does not generate any error, and tries again when another fetch access occurs.

After inserting a message into the buffer, the fetch trace state machine goes to the *running* state.

It is also possible that the fetch trace is disabled before a message is inserted into the MCS trace buffer. In this case state machine goes to the *end* state

76.5.9.2.2 Fetch trace running state

When the fetch trace state machine enters in the *running state* a counter, called CNT, is initialized to 1 and starts counting the number of sequential instruction fetch accesses.

When a fetch access with address ADDR is detected, the fetch trace logic tests if ADDR is equal to the previous fetch address, ADDR_PRV, +1 ($ADDR = ADDR_PRV + 1$). If so, no message is buffered for transmission and CNT increments by 1 ($CNT = CNT + 1$). Otherwise, if ADDR is not a sequential address, a fetch trace message is generated with the FSCNT field equals to the CNT counter value ($FSCNT = CNT$) and FADDR equals to the current address ADDR. The CNT counter is then initialized to 1.

At the moment the previous message is generated, the corresponding MCS fetch trace buffer may or may not be full. If the buffer is not full, the message is inserted into the buffer and the fetch trace state machine continues in the *running* state, otherwise a fetch trace error message is generated for the corresponding MCS and the fetch trace state machine goes to the *start* state.

Note: The MCS is a multi-thread core which executes eight threads, also called channels, in fixed time-slots of one system clock for each thread. Fetch trace is a per channel resource. Thus if more than one channel is selected for trace, each one generates its corresponding fetch trace message. There is one independent fetch state machine and CNT counter for each selected channel.

76.5.9.2.3 Fetch trace end state

In this state, the fetch trace logic ignores all fetch accesses and waits for the corresponding MCS trace buffer to be not full. When this happens, a fetch trace end message is inserted into the buffer with FSCNT equal to CNT (FSCNT = CNT). The CNT value is always reset to 0 when previous state was *start* state.

If the buffer is always full, the fetch trace logic does not exit *end* state until a reset occurs.

Note: A Fetch trace end message is sent when a system reset is detected in the GTM module. There are no further indications about the GTM reset condition.

76.5.10 Watchpoint trace

The GTMDI module provides watchpoint messaging via the auxiliary port, as defined by the IEEE-ISTO 5001-2002 standard.

76.5.10.1 Messaging

The GTMDI block provides watchpoint messaging using IEEE-ISTO 5001-2002 standard-defined public messages. When a watchpoint occurs, a watchpoint event is sent to the Event Queue. If the watchpoint condition occurs while the Event Queue is full, an error message is generated. The watchpoint message format is shown in [Figure 1939](#).

Figure 1939. Watchpoint hit message format

6 bits	K bits	15 bits
TCODE (15)	SRC	WPHIT
Length = 21+Kbits		

The values for the WPHIT packet are described in [Table 1795](#), [Table 1796](#) and [Table 1797](#). Notice that WPHIT is never 0 because the message is only generated when a watchpoint has occurred.

76.6 Initialization/application information

76.6.1 Accessing GTMDI tool-mapped registers

To initialize the JTAG port for register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Retrieve Device ID if needed
3. Load the Nexus TAP controller with the NEXUS-ENABLE instruction

To write control data to GTMDI tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select register with a pass through the SELECT-DR-SCAN path in the JTAG state machine
2. Write the register value with a pass through the SELECT-DR-SCAN path. The prior value of this register is shifted out during the write

To read status and control data from GTMDI tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear write bit to select register with a pass through SELECT-DR-SCAN path
2. Read the register value with a pass through the SELECT-DR-SCAN path. Data shifted in is ignored

See [Section 76.5.5, IEEE 1149.1 \(JTAG\) input port](#) for more detail.

77 Nexus Crossbar Multi-Master Client (NXMC)

77.1 Introduction

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2011 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters such as DMA, Ethernet and FlexRay start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools. This also provides the capability of sending user defined messages for the peripherals attached to it.

77.1.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2011 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources.
- Multi-master tracing capability for multiple master accesses
- [BLG_NXMC_0001][Covers: ADD16.091]HSM master(s) filtering from the tracing function based on external control[end]
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging
- Peripheral interface for data message transfers based on In Circuit Trace Message (ICTM) format
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on external triggers.
- Optional timestamping of the messages based on timer value and controls received at the interface
- Nexus Auxiliary port for higher data throughput
 - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
 - One $\overline{\text{EVTI}}$ (Event In) pin
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

77.2 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NAR. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$

These pins are described in [Table 1803](#).

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see [Section 77.5.1, IEEE 1149.1 \(JTAG\) Test Access Port](#) for details). A separate JTAG TAP controller is instantiated within the NXMC. [Table 1802](#) describes the JTAG pins.

Table 1802. JTAG pins for NXMC

JTAG port	Input/output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 1803. NXMC auxiliary pins

Auxiliary port	Input/output	Description
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: – one of the internal or external watchpoints has occurred and $\text{EOC} = 2'b00$ – system-level debug mode was entered (ipg_debug) and $\text{EOC} = 2'b01$ $\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (see Section 77.5.2, Enabling NXMC operation), assertion initiates data trace synchronization messages (provided data trace is enabled).

77.3 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2011 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in [Table 1804](#). The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 1804. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 1805)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 1805)
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (See Table 1806)
	14	14	ECODE	fixed	Error code (See Table 1807)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 1805)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 1804. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 1805)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (See Table 1819)
	1	26	TSTAMP	variable	Timestamp (Optional)
Peripheral Message (In circuit Trace Message: ICTM)	6	6	TCODE	fixed	TCODE number = 34
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	6	6	CKSRC	fixed	Peripheral source identifier
	1	44	CKDATA	variable	Peripheral data
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 1805. Data trace size (DSZ) encodings (TCODE = 5,6,13,14)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 1806. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved
0011	Reserved

Table 1806. Error Type (ETYPE) encodings (applicable to all error messages) (continued)

ETYPE	Description
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 1807. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description ⁽¹⁾
13 to 8	7 to 0	
Source (SRC)		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	In-Circuit Trace Message(s) Lost
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

77.4 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See [Section 77.5.1, IEEE 1149.1 \(JTAG\) Test Access Port](#) for details. The complete list of registers is shown in [Table 1808](#).

Table 1808. NXMC registers

Nexus register	Nexus access opcode	Read/ write	Read address	Write address
Development Control 1 (DC1)	\$2	R/W	\$04	\$05
Development Control 2 (DC2)	\$3	R/W	\$06	\$07
Watchpoint Trigger (WT)	\$B	R/W	\$16	\$17
Data Trace Control (DTC)	\$D	R/W	\$1A	\$1B
Data Trace Start Address1 (DTSA1)	\$E	R/W	\$1C	\$1D
Data Trace Start Address2 (DTSA2)	\$F	R/W	\$1E	\$1F
Data Trace End Address1 (DTEA1)	\$12	R/W	\$24	\$25
Data Trace End Address2 (DTEA2)	\$13	R/W	\$26	\$27
Breakpoint/Watchpoint Control Register1 (BWC1)	\$16	R/W	\$2C	\$2D
Breakpoint/Watchpoint Control Register2 (BWC2)	\$17	R/W	\$2E	\$2F
Breakpoint/Watchpoint Address Register1 (BWA1)	\$1E	R/W	\$3C	\$3D
Breakpoint/Watchpoint Address Register2 (BWA2)	\$1F	R/W	\$3E	\$3F

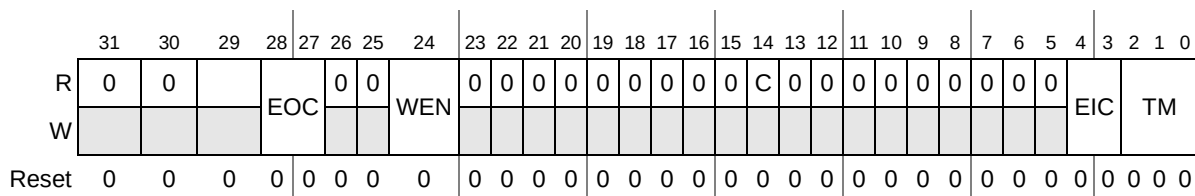
77.4.1 Register descriptions

Detailed register definitions for the current NXMC implementation are described in the following sections.

77.4.1.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. [Figure 1940](#) shows the format of the DC1.

Figure 1940. Development control register 1 (DC1)



The DC1 is described in [Table 1809](#).

Table 1809. DC1 register field description

Field	Description
31:29	Reserved for future functionality (read as 0)
28:27 EOC	EVTO Control. 00 EVTO upon occurrence of watchpoint (internal or external) 01 EVTO upon entry into system-level Debug Mode 1x Reserved
26:25	Reserved for future functionality (read as 0)

Table 1809. DC1 register field description (continued)

Field	Description
24 WEN	Watchpoint Trace Enable. 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
22:5	Reserved for future functionality (read as 0)
4:3 EIC	$\overline{\text{EVTI}}$ Control. 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled 11 Reserved
2:0 TM	Trace Mode. 000 No Trace 1xx Reserved x1x Data Trace enabled xx1 Reserved

77.4.1.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. [Figure 1941](#) shows the format of the DC2.

Figure 1941. Development control register 2 (DC2)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC2 is described in [Table 1810](#).

Table 1810. DC2 register field description

Field	Description
31:24 EWC	$\overline{\text{EVTO}}$ Watchpoint Configuration ⁽¹⁾ . 00000000 No Watchpoints trigger $\overline{\text{EVTO}}$ 1XXXXXXX External Watchpoint #1 triggers $\overline{\text{EVTO}}$ X1XXXXXX External Watchpoint #2 triggers $\overline{\text{EVTO}}$ XX1XXXXX Reserved XXX1XXXX Reserved XXXX1XXX Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$ XXXXX1XX Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$ XXXXXX1X Reserved XXXXXX1 Reserved
23:1	Reserved for future functionality (read as 0)
0 PME0	Peripheral Messaging Enable for the In-Circuit Trace (ICT) feature. ⁽²⁾ 0 Peripheral Interface disabled 1 Peripheral Interface enabled

1. The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.

2. The peripheral details are described in the device-specific chapter that describes how the modules are configured.

77.4.1.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable. The WT bits can be used to produce an address related window for triggering trace messages. [Figure 1942](#) shows the format of the WT register.

Figure 1942. Watchpoint trigger register (WT)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The WT register is described in [Table 1811](#).

Table 1811. WT register field description

Field	Description ⁽¹⁾
31:26	Reserved for future functionality (read as 0)
25:23 DTS	Data Trace Start Control. 000 Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved

Table 1811. WT register field description (continued)

Field	Description ⁽¹⁾
22:20 DTE	Data Trace End Control. 000 Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
19:0	Reserved for future functionality (read as 0)

1. The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

77.4.1.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXMC module. [Figure 1943](#) shows the format of the DTC register.

Figure 1943. Data trace control (DTC) register

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					0	0	0		0	0							0	0	0	0	0	0	0	0			0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DTC register is described in [Table 1812](#).

Table 1812. DTC field description

Field	Description
31:30 RWT1	Read/Write Trace 1. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
29:28 RWT2	Read/Write Trace 2. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
27:25	Reserved for future functionality
24 MME	Multi Master Tracing enable. 0 Tracing enabled only for Master with ID = MID (DTC[19:16]) 1 Tracing enabled for all the Masters
23:20	Reserved for future functionality

Table 1812. DTC field description (continued)

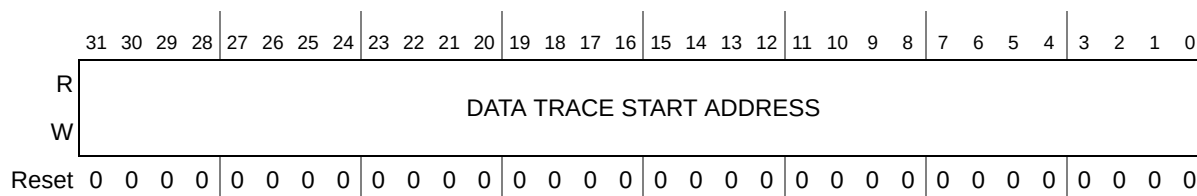
Field	Description
19:16 MID	ID of the system bus Master to be traced ⁽¹⁾
15:8	Reserved for future functionality
7 RC1	Range Control 1. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
6 RC2	Range Control 2. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
5:0	Reserved for future functionality

1. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

77.4.1.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. [Figure 1944](#) shows the format of the DTSA1 and DTSA2 registers.

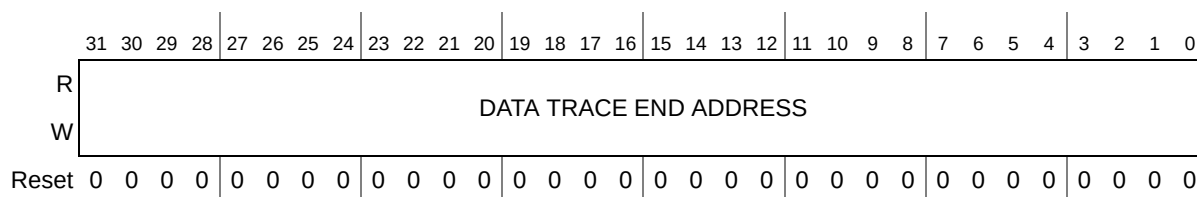
Figure 1944. Data trace start address registers 1 and 2 (DTSA1 and DTSA2)



77.4.1.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. [Figure 1945](#) shows the format of the DTEA1 and DTEA2 registers.

Figure 1945. Data trace end address registers 1 and 2 (DTEA1 and DTEA2)



[Table 1813](#) illustrates the range that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 1813. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA

Table 1813. Data trace address range options (continued)

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range—no trace

Note: DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.

When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

77.4.1.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. [Figure 1946](#) shows the format of the BWC1 register.

Figure 1946. Breakpoint/watchpoint control register 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1		BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The BWC1 register is described in [Table 1814](#).

Table 1814. BWC1 field description

Field	Description
31:30 BWE1	Breakpoint/Watchpoint #1 Enable. 00 Internal Nexus Watchpoint #1 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #1 enabled
29:28 BRW1	Breakpoint/Watchpoint #1 Read/Write Select. 00 Watchpoint #1 hit on read accesses 01 Watchpoint #1 hit on write accesses 10 Watchpoint #1 on read or write accesses 11 Reserved
27:18	Reserved for future functionality (read as 0)
17:16 BWR1	Breakpoint/Watchpoint #1 Register Compare. 00 No Register Compare (same as BWC1[31:30] = 2'b00) 01 Reserved 10 Compare with BWA1 value 11 Reserved

Table 1814. BWC1 field description (continued)

Field	Description
15 BWT1	Breakpoint/Watchpoint #1 Type. 0 Reserved 1 Watchpoint #1 on data accesses
14:0	Reserved for future functionality (read as 0)

77.4.1.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. [Figure 1947](#) shows the format of the BWC2 register.

Figure 1947. Breakpoint/watchpoint control register 2

		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	W	BWE2	BRW2	0	0	0	0	0	0	0	0	0	0	BWR2	BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The BWC2 register is described in [Table 1815](#).

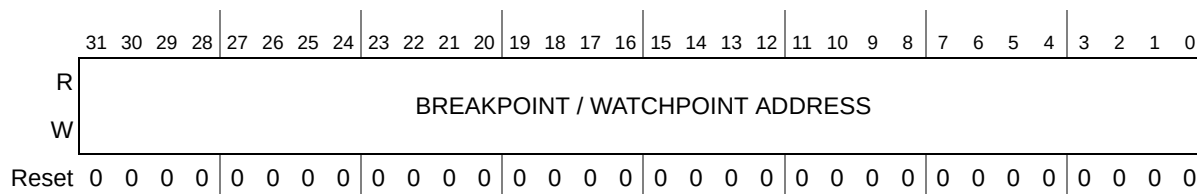
Table 1815. BWC2 field description

Field	Description
31:30 BWE2	Breakpoint/Watchpoint #2 Enable. 00 Internal Nexus Watchpoint #2 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #2 enabled
29:28 BRW2	Breakpoint/Watchpoint #2 Read/Write Select. 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
27:18	Reserved for future functionality (read as 0)
17:16 BWR2	Breakpoint/Watchpoint #2 Register Compare. 00 No Register Compare (same as BWC2[31:30] = 2b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved
15 BWT2	Breakpoint/Watchpoint #2 Type. 0 Reserved 1 Watchpoint #2 on data accesses
14:0	Reserved for future functionality (read as 0)

77.4.1.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. [Figure 1948](#) shows the format of the BWA1 and BWA2 registers.

Figure 1948. Breakpoint /watchpoint address registers 1 and 2



77.4.1.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in [Table 1808](#). For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

77.5 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of $\overline{\text{TRST}}$ and system reset de-assertion, after the JTAG ID register has been read by the tool.

77.5.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in [Section 77.5.4, NXMC register access via JTAG](#).

77.5.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

77.5.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain

information about the Nexus module it is accessing, such as version, sequence, feature set, etc.

77.5.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in [Table 1808](#).

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see [Table 1808](#)), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown in [Figure 1949](#) and [Table 1816](#).

Figure 1949. JTAG DR for Nexus register access

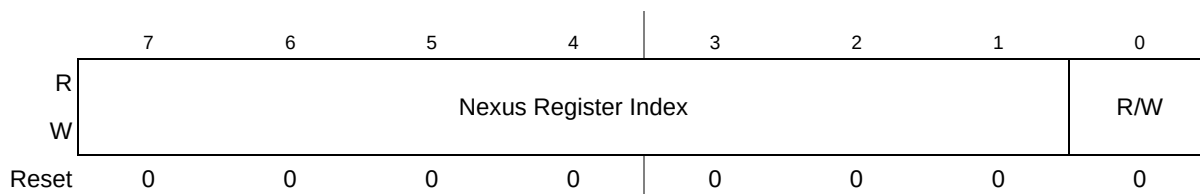


Table 1816. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in Table 1808
Read/Write (R/W)	0Read 1Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
 - a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
 - b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

77.6 Functional description

77.6.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2011 standard.

77.6.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

77.6.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

77.6.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32, 64. The TSTAMP field is 26 bits. Data write message and data read message information is messaged out in the format shown in [Figure 1950](#). The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Figure 1950. DTM message format for DATA width = 64 bits

8/16/32/64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010 or 111011)

TSTAMP is another optional field which is inserted based on the controls received from the NAR, see [Section 77.6.4, Timestamping feature](#).

77.6.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

77.6.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in [Figure 1951](#). The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Figure 1951. DTM error message format (fixed length = 48 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

77.6.1.2.4 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see [Table 1817](#)):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message.
- Upon returning from a low power state, the first data trace message is a synchronization message.
- Upon returning from debug mode, the first data trace message is a synchronization message.
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message.
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred.
- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature.
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message. This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic.

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in [Figure 1952](#). The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Figure 1952. Data write/read with synchronization message format

8-64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100 or 111101)

Exception conditions that result in data trace synchronization are summarized in [Table 1817](#).

Table 1817. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).
Collision Priority	All messages have the following priority: Error > WPM > ICTM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

77.6.1.3 DTM operation

77.6.1.3.1 Enabling data trace messaging

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).
- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. (The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note: When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (i.e. DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure correct DTM functionality.

77.6.1.4 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

Note: If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority ($WPM > ICTM > DTM$).

77.6.1.5 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2011 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address ($A1$) = 0x0003FC01, the new address ($A2$) = 0x0003F365 as shown in [Figure 1953](#).

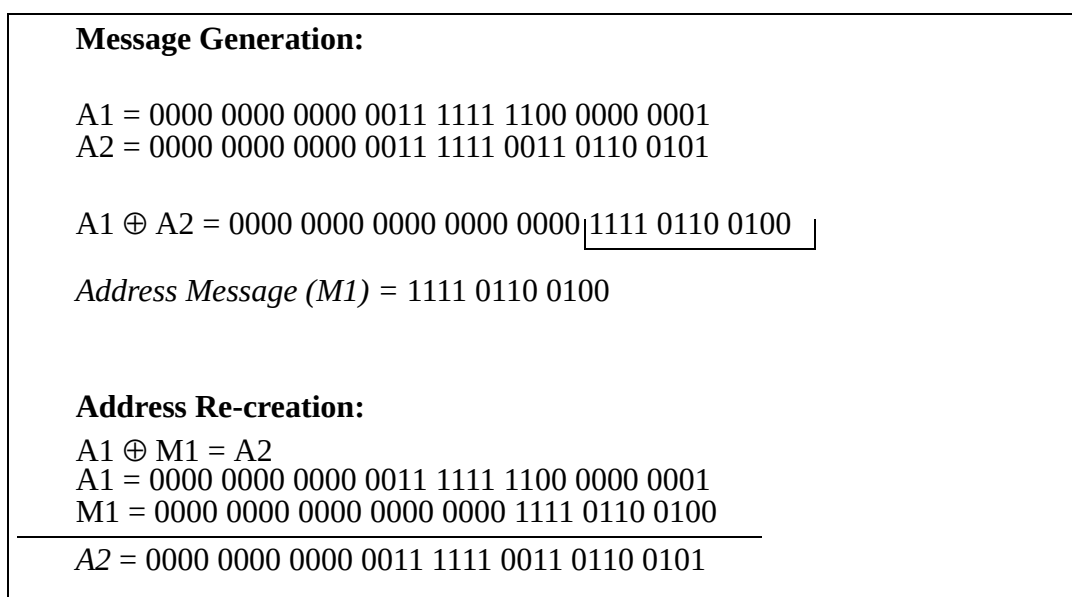


Figure 1953. Relative addressing example

77.6.1.6 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

77.6.1.7 System bus cycle special cases

The system bus cycle special cases are shown in [Table 1818](#).

Table 1818. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

77.6.1.8 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (see figure in [Section 77.6.1.2.1, Data write and data read messages](#) for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

77.6.1.9 HSM master filtering

[BLG_NXMC_0004][Covers: ADD16.091]For security reasons certain HSM masters need to be disabled or filtered from data tracing operations.[end]

77.6.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2011. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

77.6.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note: In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (i.e. BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints—Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in [Section 77.6.1.8, System bus multi-master tracing](#)).
- External watchpoints—The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level. The watchpoints are described in the device-specific chapter that describes how

the modules are configured. If either of these signals asserts (i.e, on 0-to-1 transition), a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in [Figure 1954](#) and [Table 1819](#).

Figure 1954. Watchpoint message format (fixed length = 18 bits)

8 bits	4 bits	6 bits
WPHIT (RRRRXXXX)	SRC	TCODE (001111)

Table 1819. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1 ⁽¹⁾	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

1. R = reserved for future use (insert 0s).

77.6.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in [Figure 1955](#).

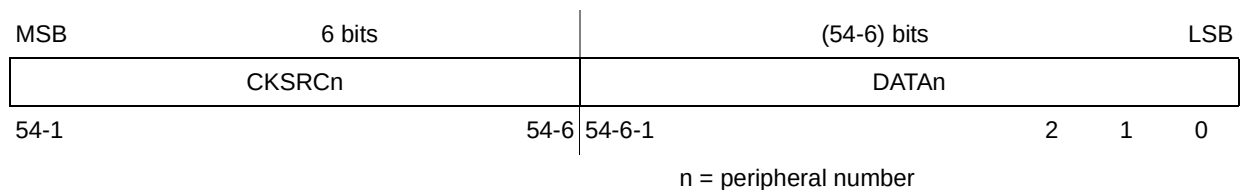
Figure 1955. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

77.6.3 Peripheral interface

The NXMC provides an asynchronous interface to peripherals for sending data messages. The NXMC can back pressure or delay the data transfer using a data accept signal. A round-robin scheme is followed between the peripherals for forwarding received data to the queue; therefore, each peripheral has an equal priority and hence bandwidth for data transfers.

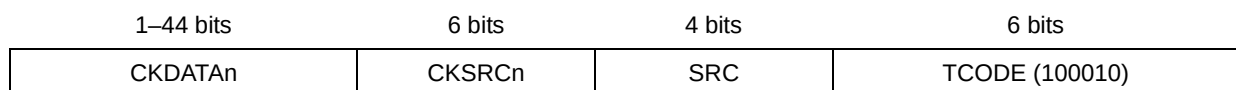
To enable messaging from peripherals there are controls present in the DC2 register for individual peripherals: DC2[PME0]. There are no message generated and hence data accepted from a peripheral unless the corresponding enable bit is set to 1.

Figure 1956. Peripheral DATA bus fields

Another input port is present which indicates data loss inside the peripheral itself due to over-run, the module in turn generates error message to communicate such events to the debugger.

77.6.3.1 Peripheral data message format (ICTM)

The message formats for the peripheral data messages (formatted as ICTM) is shown in [Figure 1957](#). The SRC field values are described in the device-specific chapter that describes how the modules are configured. The CKSRC field is received from the MSB 6 bits of the individual peripheral data bus, and CKDATA is a variable field and contains 44 LSBits received on the databus of the peripheral interface.

Figure 1957. Peripheral message format (max length = 60 bits, min length = 17 bits)

77.6.3.2 Peripheral error message format

The error messages are generated based data over-run indication from a peripheral. Error information is messaged out in the format shown in [Figure 1958](#).

Figure 1958. Error message format (fixed length = 28 bits)

The SRC field values are described in the device-specific chapter that describes how the modules are configured. ETYPE and ECODE encoding are listed in [Table 1806](#) and [Table 1807](#).

77.6.4 Timestamping feature

This feature allows a timestamp packet to be appended to all messages, based on the timer value and controls received externally from the NAR. If enabled, this packet is added as the last field to all the outgoing messages. The TSTAMP packet width is 26 bits.

The TSTAMP packet is appended at the end of a message during message formatting, before transfer. [Figure 1959](#) shows an example of ICTM message with a TSTAMP field.

Figure 1959. Message with TSTAMP field example

26 bits	1–44 bits	6 bits	4 bits	6 bits
TSTAMP	CKDATAn	CKSRCn	SRC	TCODE (100010)

The enable or disable indication should remain static for the complete debug session. Dynamically enabling or disabling of the timestamping is not allowed during running debug tracing. The following steps allow seamless enabling/disabling of the timestamping:

- 1. All tracing (DTM/WPM/ICTM) must be disabled
- 2. Wait for the internal queues to be emptied (no further request is asserted at the auxiliary interface)
- 3. Change the timestamp enable indication to assert/deassert
- 4. Enable the required tracing controls (via TM, PME0 etc. control bits)



78 Cyclic Redundancy Check (CRC) Unit

78.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading this work from the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of three hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

78.2 Main features

- 1 context(s) for the concurrent CRC computation
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- 3 hard-wired polynomials (CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)
- Support for byte, halfword, or word width of the input data stream

78.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT
- CRC-32 Ethernet

78.3 Block diagram

The top level diagram of the CRC module is given in [Figure 1960](#). Refer to the chip configuration chapter for number of contexts in this module.

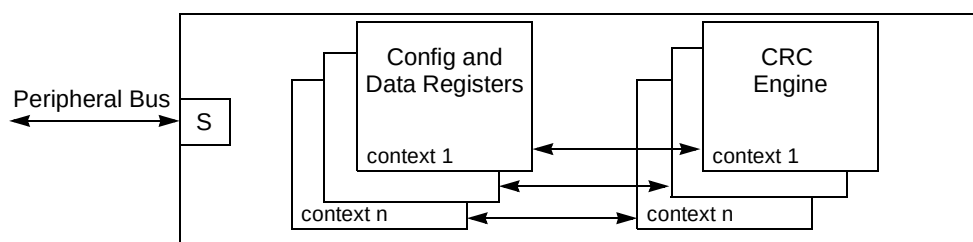


Figure 1960. CRC top level diagram

78.4 Signal description

The CRC module does not generate any external signals.

78.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers

Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are accessible (read/write) in each access mode: user, supervisor, and test.

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

78.5 Memory map and registers

This section describes, in address-order, all the CRC registers.

78.5.1 Register description

[Table 1820](#) shows the memory map for the CRC registers.

Table 1820. CRC unit memory map⁽¹⁾

Offset from CRCU_BASE	Register	Register classification ⁽²⁾	Location
000h + (N-1)*10h	CRC Configuration Register (CRC_CFG)	Saf-Lat	on page 3423
004h + (N-1)*10h	CRC Input Register (CRC_INP)	Saf-Relv	on page 3424

Table 1820. CRC unit memory map⁽¹⁾ (continued)

Offset from CRCU_BASE	Register	Register classification ⁽²⁾	Location
008h + (N-1)*10h	CRC Current Status Register (CRC_CSTAT)	Saf-Relv	on page 3425
00Ch + (N-1)*10h	CRC Output Register (CRC_OUTP)	Saf-Relv	on page 3426

1. Where N = context number. Refer to the chip configuration chapter for number of contexts used in this application.

2. No-Saf = Not safety relevant, Saf-Lat = Safety Latent, Saf-Relv = Safety Relevant, Saf-Crit = Safety Critical.

78.5.1.1 CRC Configuration Register (CRC_CFG)

Figure 1961. CRC Configuration Register (CRC_CFG)

Address : Base + 000h + (N-1)*10h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0						
W											SWAP_BYTEWISE	SWAP_BITWISE	POLYG		SWAP	INV
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	f(N) (1)	0	0

1. Reset value is a function of N. When N%2 = 0, reset value = 1. When N%2 = 1, reset value = 0.

Table 1821. CRC_CFG field descriptions

Field	Description
SWAP_BYTEWISE	SWAP_BYTEWISE 0 No swap 1 Byte wise (MSB ↔ LSB bytes) swap performed on CRC_INP input data internally for CRC-32 polynomial calculations. <i>Note: INV and SWAP bits are set to '1' for CRC-32 polynomial calculations.</i>
SWAP_BITWISE	SWAP_BITWISE 0 No swap 1 Bit wise (MSB ↔ LSB) bits swap performed on CRC_INP input data internally for CRC-16 polynomial calculations.

Table 1821. CRC_CFG field descriptions (continued)

Field	Description
POLYG	Polynomial selection 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 Reserved This bit can be read and written by software. <i>Note: This bit can be written only during the configuration phase.</i>
SWAP	SWAP selection 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB → LSB, LSB → MSB) applied on the CRC_OUTP content. In case of CRC-CCITT polynomial, the swap operation is applied on the 16 LSB bits. This bit can be read and written by software. <i>Note: This bit can be written only during the configuration phase.</i>
INV	INV selection 0 No inversion selection applied on the CRC_OUTP content. 1 Inversion selection (bit x bit) applied on the CRC_OUTP content. In case of CRC-CCITT polynomial the inversion operation is applied on the 16 LSB bits. This bit can be read and written by software. <i>Note: This bit can be written only during the configuration phase.</i>

78.5.1.2 CRC Input Register (CRC_INP)

Figure 1962. CRC Input Register (CRC_INP)

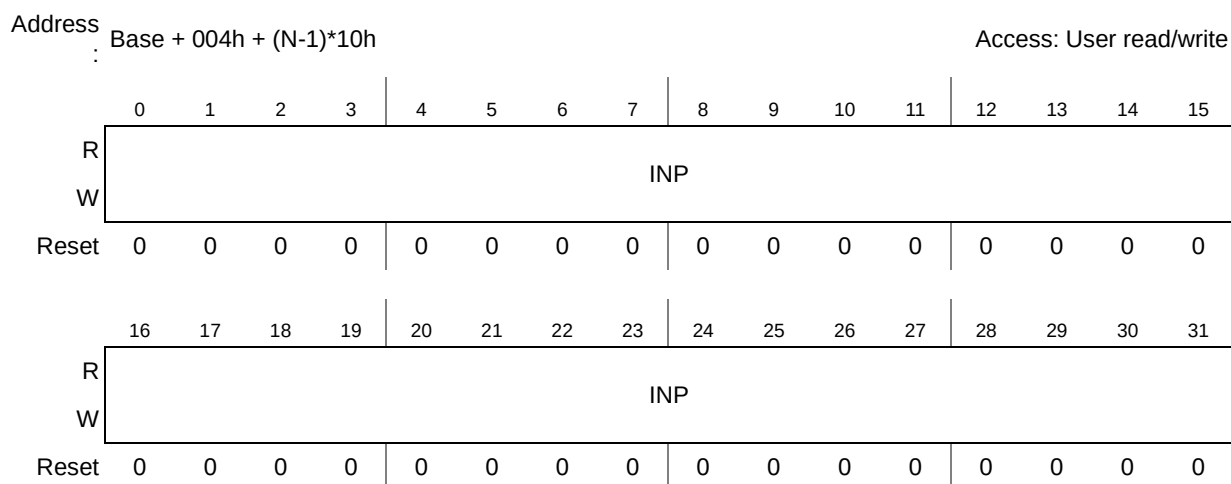


Table 1822. CRC_INP field descriptions

Field	Description
INP	Input data for the CRC computation The INP register can be written at byte, halfword (high and low), or word in any sequence. In case of halfword write operation, the bytes must be contiguous. Only the bits written are fed to the CRC engine. This register can be read and written by software.

78.5.1.3 CRC Current Status Register (CRC_CSTAT)

Figure 1963. CRC Current Status Register (CRC_CSTAT)

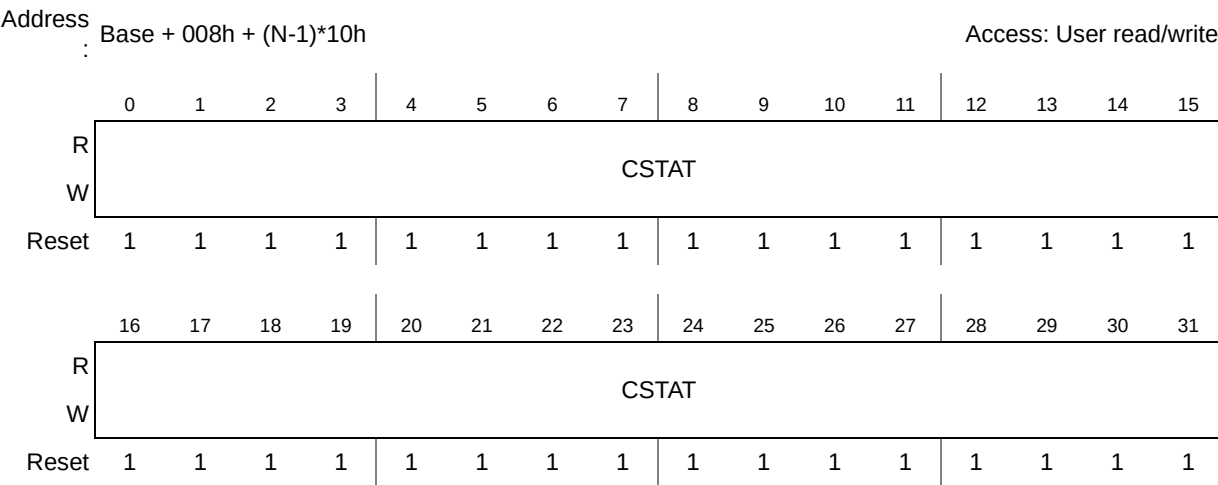


Table 1823. CRC_CSTAT field descriptions

Field	Description
CSTAT	Status of the CRC signature The CSTAT register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the16 LSB bits are significative. The 16 MSB bits are tied at 0b during the computation. In the case of the CRC-8 polynomial, only the 8 LSB bits are significative. The 24 MSB bits are tied at 0b during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be read and written by software. <i>Note: This register can be written only during the configuration phase.</i>

78.5.1.4 CRC Output Register (CRC_OUTP)

Figure 1964. CRC Output Register (CRC_OUTP)

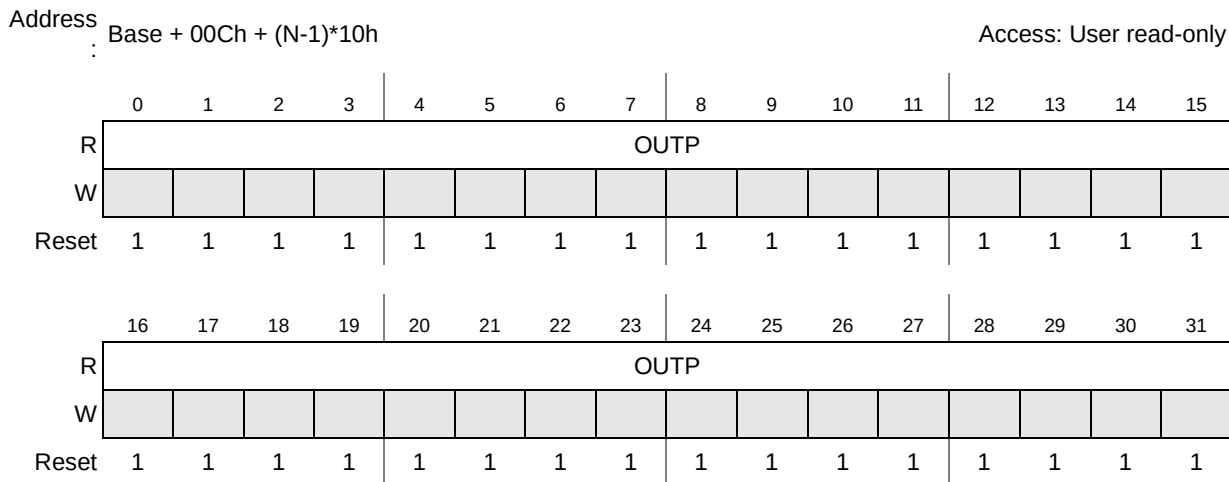


Table 1824. CRC_OUTP field descriptions

Field	Description
OUTP	Final CRC signature The OUTP register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the 16 LSB bits are significative. The 16 MSB bits are tied at 0b during the computation. In the case of the CRC-8 polynomial, only the 8 LSB bits are significative. The 24 MSB bits are tied at 0b during the computation. This register can be read by software.

78.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by n concurrent DMA data transfers (mem2mem) where n is less than or equal to the number of contexts.

- The standard generator polynomials are given in [Equation 88](#), [Equation 89](#), and [Equation 90](#) for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

Equation 88 CRC8 VDA CAN:

$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 89 CRC-CCITT (x.25 protocol):

$$x^{16} + x^{12} + x^5 + 1$$

Equation 90 CRC-32 (Ethernet protocol):

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

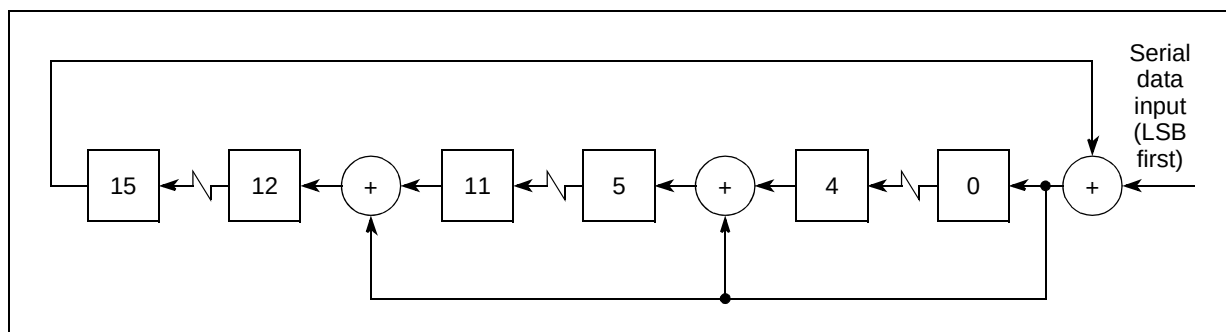


Figure 1965. CRC-CCITT engine concept scheme

The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. The concept scheme (serial data loading) of the CRC engine is given in [Figure 1965](#) for the CRC-CCITT. The design implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the peripheral bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (for example, SPI, etc.), a bit swap (MSB → LSB, LSB → MSB) and/or bit inversion of the final CRC signature can be applied (CRC_OUTP register) before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in [Figure 1966](#).

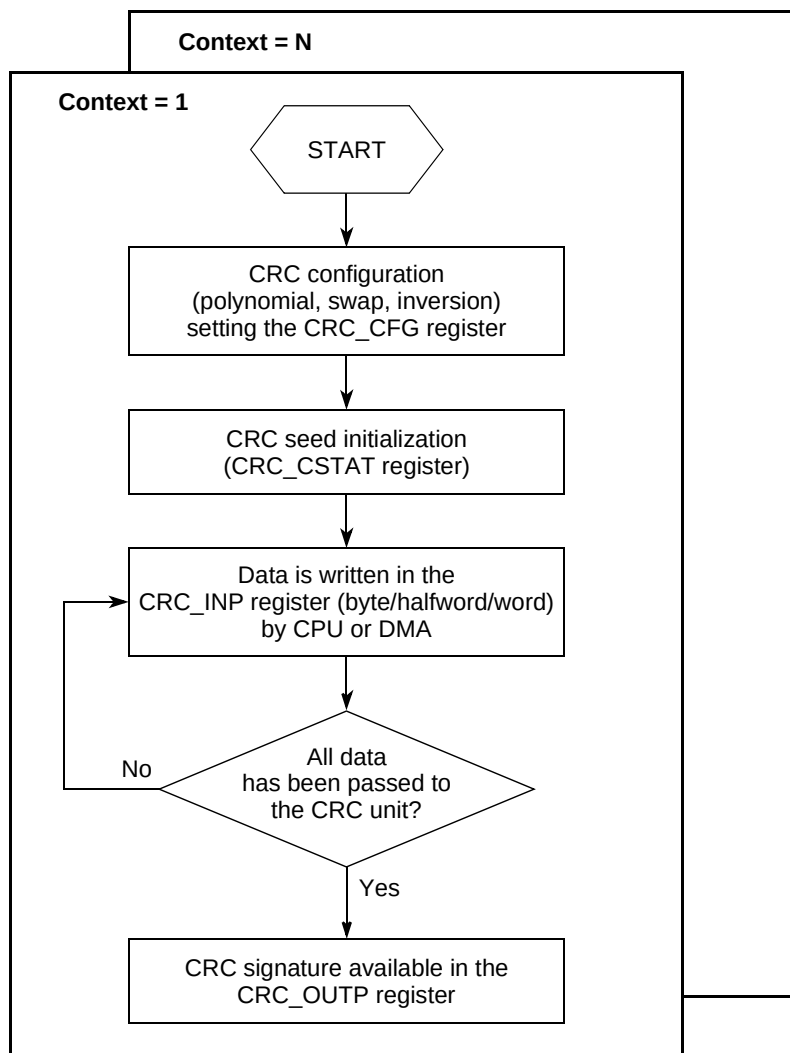


Figure 1966. CRC computation flow

78.7 Use cases

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has n channels (greater than or equal to the number of contexts) configurable for the following types of data transfer — mem2mem, periph2mem, mem2periph — then this sequence, as shown in [Figure 1967](#), is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x , channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB → LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

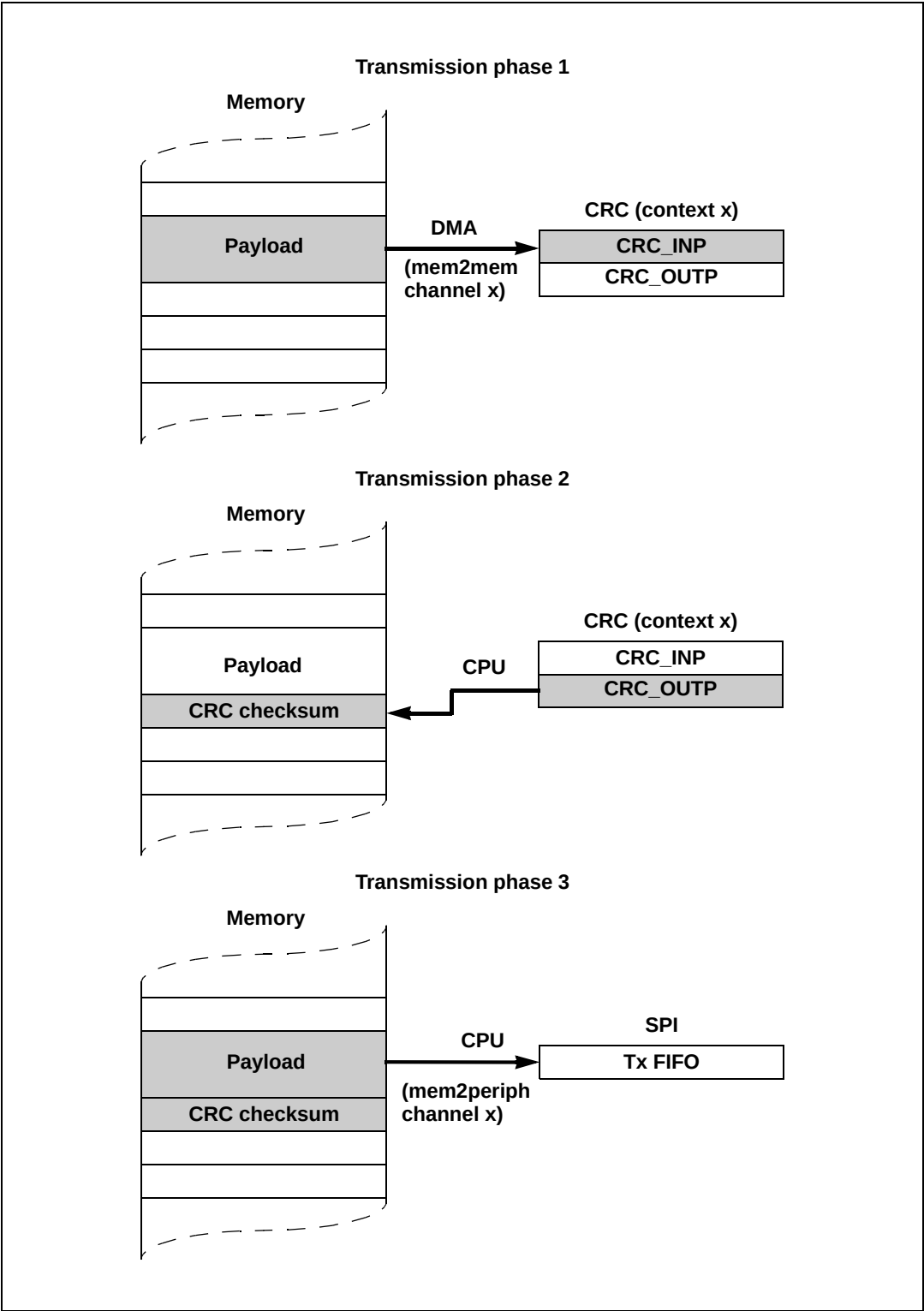


Figure 1967. DMA-CRC transmission sequence

79 Memory Error Management Unit (MEMU)

79.1 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on SRAM, Peripheral System RAM and flash memory. When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
 - Single Bit error in the data part that is detected via ECC for a System RAM or Peripheral RAM or flash memory.
 - Single Bit error in the data part that is detected via MBIST on any RAM.
- Uncorrectable Error: It comprises the following:
 - Multiple bit error that is detected via ECC for a Peripheral SRAM, System RAM or flash memory.
 - Multiple bit error that is detected via MBIST on any SRAM. Based on MBIST implementation, the MBIST might report several correctable errors in the same word instead of an uncorrectable one. It is the task of the software after MBIST to aggregate these reports and detect the uncorrectable error.
 - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

When multiple errors are indicated from various sources at same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from either ECC or MBIST logic), independent of the source generating these errors.

The MEMU has three instances of the basic reporting block; one each for System RAM ECC and MBIST, Peripheral SRAM ECC or flash memory ECC.

Each instance has two reporting tables—Correctable errors are reported in one table while the uncorrectable are reported in the other.

See [Section 79.4, Design overview](#) for the reporting block details.

79.2 Features

The MEMU has the following features:

- Supports up to 128 error sources per Reporting Block (System RAM ECC and MBIST, Peripheral SRAM ECC or flash memory ECC): The number of error sources are chip-

specific; see the Device Configuration chapter that describes how modules are configured and connected.

- Support for error reporting from the following category of error sources (both for ECC and MBIST):
 - System RAM
 - Peripheral SRAM
 - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.
- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.
- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

79.3 Block diagram

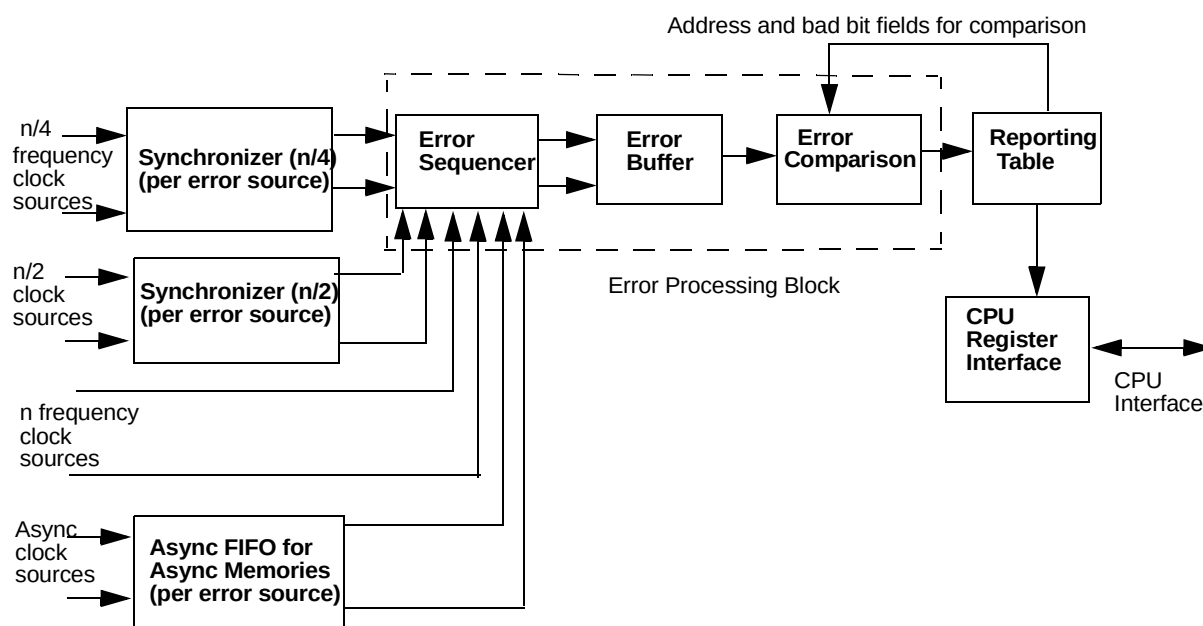


Figure 1968. MEMU block diagram

79.4 Design overview

The MEMU block has three instantiations of the basic reporting block along with a common CPU interface for the user software to access the storage block.

The MEMU should be enabled on power on reset. The user software can write known error addresses into the reporting table to prevent reporting of those error addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See [Section 79.7.3, Handling overflows \(Multiple error reporting\)](#) for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits) are concatenated and stored in the FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

2. Error Processing block

This block is used to process the incoming errors coming from the device.

- Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.
- All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.
- While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit (depending on ECC or MBIST logic) of the incoming error with the one stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set., else the incoming error is dropped.
- While doing the error buffer uniqueness check, the type of error i.e Correctable Error (CE) or Uncorrectable Error (UCE) is also compared, hence even if the Error Address and the Bad bit fields (Depending on whether the incoming error is generated from ECC or MBIST logic) of the incoming error matches with the error buffer, if the type of errors (CE and UCE) differ they are treated as unique
- Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable)
- Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (≤ 5 unless the error gets temporarily stored in the error

buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

The CPU can clear the flags signalling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (i.e. [Section 79.6.2.4, System RAM correctable error reporting table status register \(SYS_RAM_CERR_STS_n\)](#) or [Section 79.6.2.6, System RAM uncorrectable error reporting table status register \(SYS_RAM_UNCERR_STS\)](#)). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

The reporting table sizes are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

4. Register block

The CPU register programming interface comprises of memory mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signalled to the CPU and the CPU command will be completed in the next clock cycle.

The software, in principle, is not supposed to write VLD to 1 if it is already 1.

The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

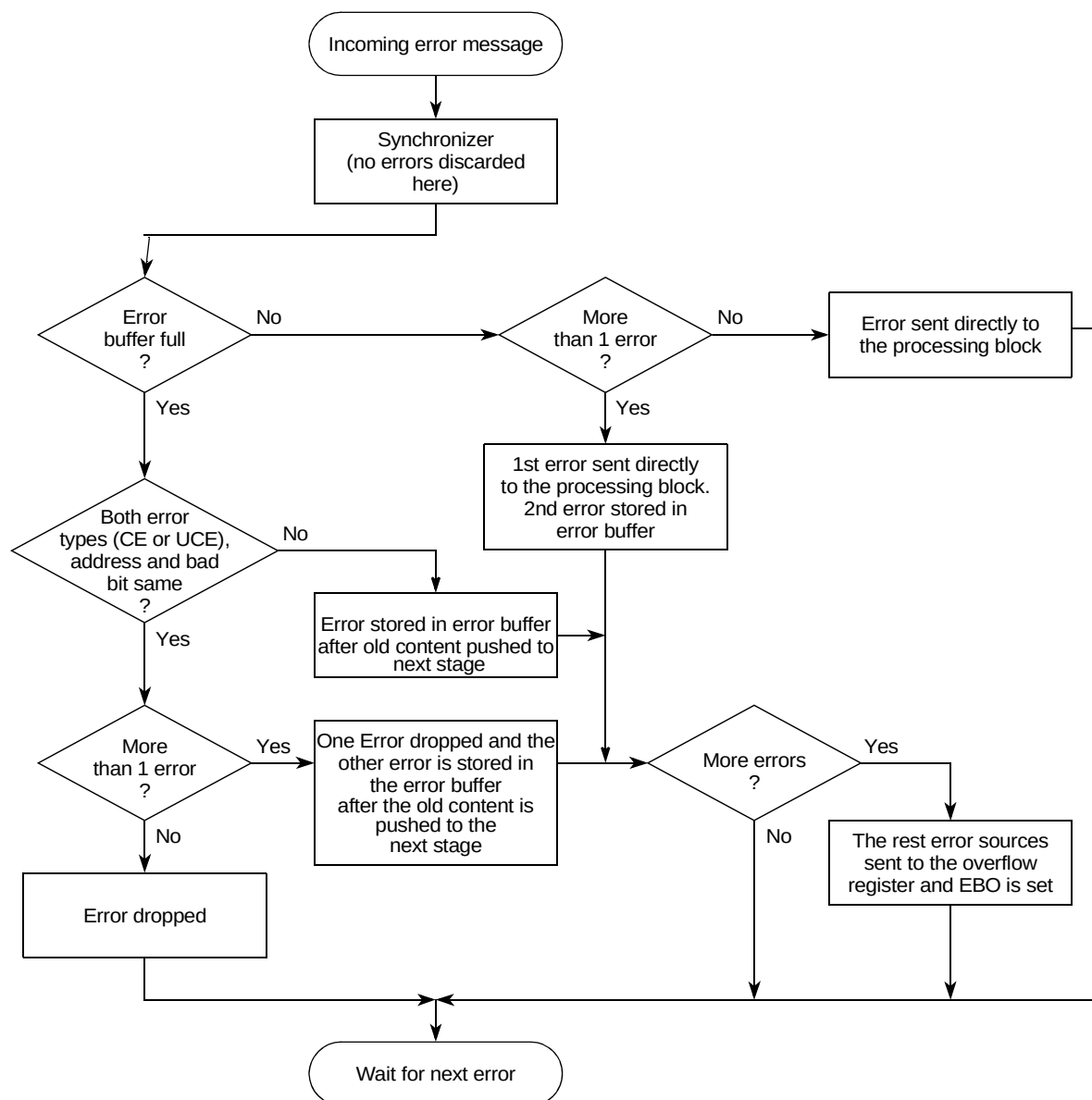


Figure 1969. MEMU error buffer uniqueness check flow chart

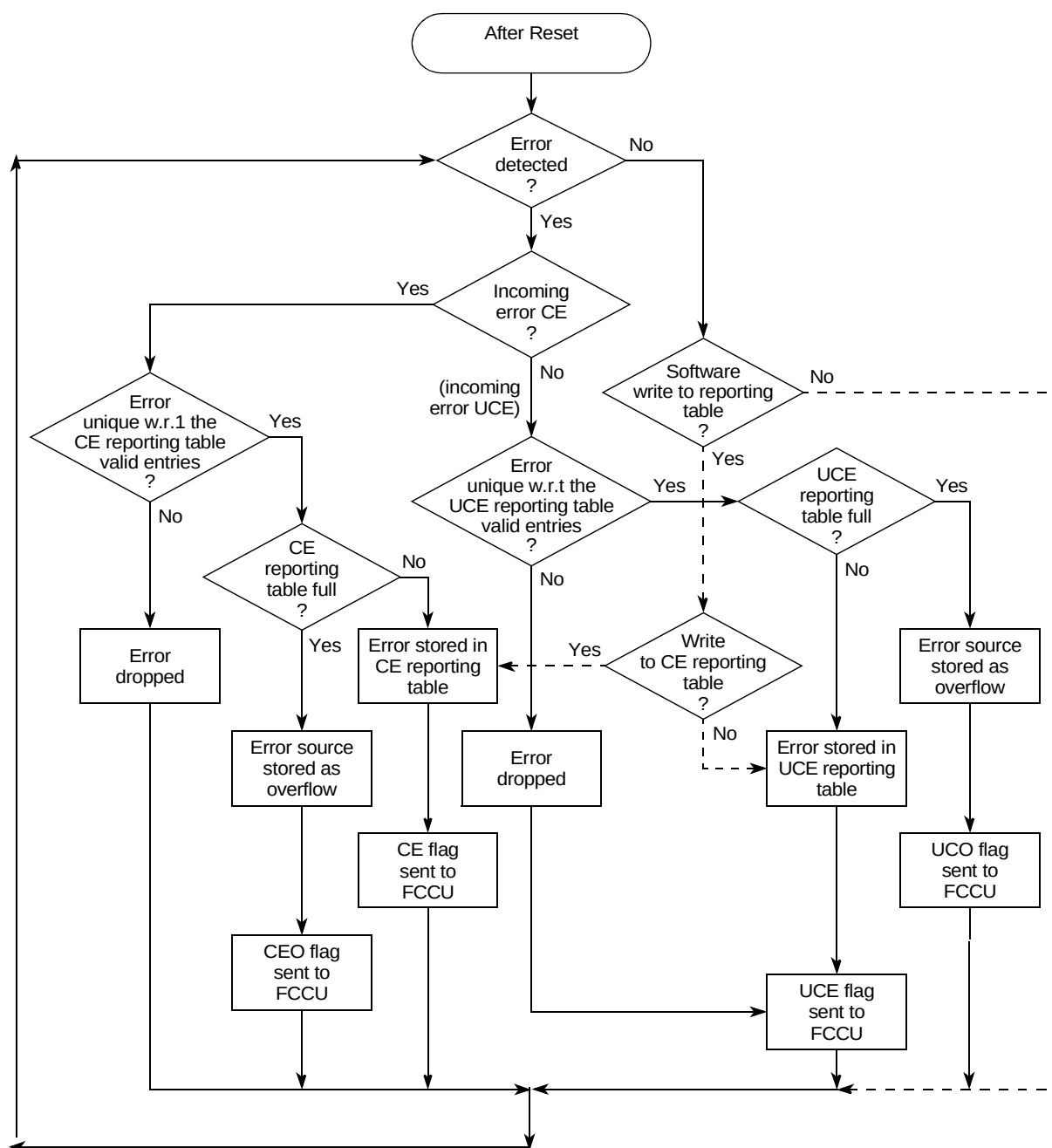


Figure 1970. MEMU reporting table uniqueness check flow chart

79.5 External signal description

MEMU has no external signals.

79.6 Memory map and register definition

79.6.1 Memory map

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

Note: The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

The Control, Error Flag and Debug registers are common for all MEMU error reporting blocks. The correctable error status, correctable error address, uncorrectable error status, uncorrectable error address and concurrent error overflow registers are replicated for each MEMU instance.

Note: The actual availability, number of registers and number of reported errors in the reporting table are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

Table 1825. MEMU memory map

Offset or address	Register	Access	Reset value	Section/ page
0x0000	Control register (CTRL)	R/W	0000_0000	79.6.2.1/3439
0x0004	Error flag register (ERR_FLAG)	R/W	0000_0000	79.6.2.2/3439
0x000C	Debug register (DEBUG)	R/W	0000_0000	79.6.2.3/3442
0x0020 + n × 0x8 (n = 0 to 10–1)	System RAM correctable error reporting table status registers (SYS_RAM_CERR_STS _n)	R/W	0000_0000	79.6.2.4/3444
0x0024 + n × 0x8 (n = 0 to 10–1)	System RAM correctable error reporting table address registers (SYS_RAM_CERR_ADDR _n)	R/W	0000_0000	79.6.2.5/3445
0x0028 + (10 – 1) × 0x8	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)	R/W	0000_0000	79.6.2.6/3445
0x002C + (10 – 1) × 0x8	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)	R/W	0000_0000	79.6.2.7/3446
0x0030 + n × 0x4 + (10 – 1) × 0x8 (n = 0 to 94 ÷ 32 – 1) ⁽¹⁾	System RAM concurrent overflow registers ⁽²⁾ (SYS_RAM_OFLW _n)	R/W	0000_0000	79.6.2.8/3446
0x0620 + n × 0x8 (n = 0 to 2–1)	Peripheral RAM correctable error reporting table status registers (PERIPH_RAM_CERR_STS _n)	R/W	0000_0000	79.6.2.9/3447

Table 1825. MEMU memory map (continued)

Offset or address	Register	Access	Reset value	Section/ page
0x0624 + n × 0x8 (n = 0 to 2–1)	Peripheral RAM correctable error reporting table address registers (PERIPH_RAM_CERR_ADDRn)	R/W	0000_0000	79.6.2.10/3448
0x0628 + (2–1) × 8	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)	R/W	0000_0000	79.6.2.11/3449
0x062C + (2–1) × 8	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)	R/W	0000_0000	79.6.2.12/3450
0x0630 + n × 0x4 + (2–1) × 0x8 (n = 0 to 25÷32 – 1) ⁽¹⁾	Peripheral RAM concurrent overflow registers ¹ (PERIPH_RAM_OFLWn)	R/W	0000_0000	79.6.2.13/3450
0x0C20 + n × 0x8 (n = 0 to 20–1)	Flash memory correctable error reporting table status registers (FLASH_CERR_STS _n)	R/W	0000_0000	79.6.2.14/3451
0x0C24 + n × 0x8 (n = 0 to 20–1)	Flash memory correctable error reporting table address registers (FLASH_CERR_ADDRn)	R/W	0000_0000	79.6.2.15/3452
0x0C28 + (20–1) × 8	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)	R/W	0000_0000	79.6.2.16/3452
0x0C2C + (20–1) × 8	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)	R/W	0000_0000	79.6.2.17/3453
0x0C30 + n × 0x4 + (20–1) × 0x8 (n = 0 to 3÷32 – 1) ⁽¹⁾	Flash memory concurrent overflow registers ¹ (FLASH_OFLWn)	R/W	0000_0000	79.6.2.18/3454

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is $94 \div 32 = 2.9375$ or $66 \div 32 = 2.0625$, round the result to 3.
2. The corresponding error sources for the Concurrent Overflow Register (OFLW) are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

79.6.2 Register descriptions

79.6.2.1 Control register (CTRL)

Figure 1971. Control register (CTRL)

Offset 0x0000

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SWR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1826. CTRL field descriptions

Field	Description
0–15	Reserved
16 SWR	Software Reset bit Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.
17–31	Reserved

79.6.2.2 Error flag register (ERR_FLAG)

Bits in the ERR_FLG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a '1'. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Figure 1972. Error flag register (ERR_FLAG)

Offset 0x0004

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	PR_CE	PR_UCE	PR_CEO	PR_UCO	PR_EBO
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	F_CE	F_UCE	F_CEO	F_UCO	F_EBO	0	0	0	SR_CE	SR_UCE	SR_CEO	SR_UCO	SR_EBO
W				w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Table 1827. ERR_FLAG field descriptions

Field	Description
0–10	Reserved
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag. Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module) 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.

Table 1827. ERR_FLAG field descriptions (continued)

Field	Description
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
16–18	Reserved
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: – a new Flash ECC uncorrectable error was detected. – a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
24–26	Reserved
27 SR_CE	System RAM ECC and MBIST Correctable Error detect flag. Indicates that a new and unique System RAM ECC and MBIST correctable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.

Table 1827. ERR_FLAG field descriptions (continued)

Field	Description
28 SR_UCE	System RAM ECC and MBIST Uncorrectable Error Detect flag. Asserted when: – a new System RAM ECC and MBIST uncorrectable error is detected. – a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
29 SR_CEO	System RAM ECC and MBIST Correctable error Overflow flag. Asserted when the System RAM ECC and MBIST correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
30 SR_UCO	System RAM ECC and MBIST Uncorrectable error Overflow flag. Asserted when the System RAM ECC and MBIST uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECC and MBIST Error buffer Overflow. Flag asserted when the internal error processing buffer of System RAM ECC and MBIST MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

79.6.2.3 Debug register (DEBUG)

Figure 1973. Debug register (DEBUG)

Offset 0x000C												Access: R/W				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	FR_PR_CE	FR_PR_UCE	FR_PR_CEO	FR_PR_UCO	FR_PR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	FR_F_CE	FR_F_UCE	FR_F_CEO	FR_F_UCO	FR_F_EBO	0	0	0	FR_SR_CE	FR_SR_UCE	FR_SR_CEO	FR_SR_UCO	FR_SR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1828. DEBUG field descriptions

Field	Description
0–10	Reserved
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag to 1.
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag to 1.
16–18	Reserved
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag to 1.
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag to 1.
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag to 1.
24–26	Reserved
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag to 1.
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag to 1.

Table 1828. DEBUG field descriptions (continued)

Field	Description
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag to 1.

79.6.2.4 System RAM correctable error reporting table status register (SYS_RAM_CERR_STS_n)

Note: The reporting table is not cleared on software reset.

Figure 1974. System RAM correctable error reporting table status register (SYS_RAM_CERR_STS_n)

Offset: 0x0020 + n × 0x8 (n = 0 to 10–1)

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1829. SYS_RAM_CERR_STS_n field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23	Reserved
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. For non-BIST type of errors, this field is not used and reads as 0xFF Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

79.6.2.5 System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDRn)

Figure 1975. System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDRn)

Offset: $0x0024 + n \times 0x8$ ($n = 0$ to $10-1$)

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1830. SYS_RAM_CERR_ADDRn field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

79.6.2.6 System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Figure 1976. System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Offset: $0x0028 + (10 - 1) \times 0x8$

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1831. SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31	Reserved

79.6.2.7 System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Figure 1977. System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Offset: 0x002c + (10–1) × 0x8

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1832. SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This register must be READ ONLY when its corresponding valid bit is not asserted.

79.6.2.8 System RAM concurrent overflow registers (SYS_RAM_OFLWn)

For example, if there are 32 channels for a specific MEMU instance and the size of the reporting table is 10 then there will be 10 correctable error reporting table status and address registers, 1 uncorrectable error reporting table status and address register and 1 overflow register. However, if the number of channels are increased to 96, the overflow registers will increase to 3 as per the formula below.

Note: The reporting table is not cleared on software reset.

Figure 1978. System RAM concurrent overflow registers (SYS_RAM_OFLWn)

Offset: $0x0030 + n \times 0x4 + (10-1) \times 0x8$
 $(n = 0 \text{ to } 94 \div 32 - 1)^{(1)}$

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW[31:16]															
W	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW[15:0]															
W	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is $94 \div 32 = 2.9375$ or $66 \div 32 = 2.0625$, round the result to 3.

Table 1833. SYS_RAM_OFLWn field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Section 79.7.3, Handling overflows (Multiple error reporting) , occurs.

79.6.2.9 Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STSn)

Note: The reporting table is not cleared on software reset.

Figure 1979. Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STSn)

Offset: $0x0620 + n \times 0x8$ ($n = 0 \text{ to } 2-1$)

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1834. PERIPH_RAM_CERR_STS_n field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23	Reserved
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. For non-BIST type of errors, this field is not used and reads as 0xFF Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

79.6.2.10 Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR_n)

Figure 1980. Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR_n)

Offset: 0x0624 + n × 0x8 (n = 0 to 2–1)

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1835. PERIPH_RAM_CERR_ADDR_n field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

79.6.2.11 Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Figure 1981. Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Offset: 0x0628 + (2–1) × 8

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1836. PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31	Reserved

79.6.2.12 Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Figure 1982. Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Offset: 0x062C + (2−1) × 8																Access: R/W			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	ERR_ADD[31:16]																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	ERR_ADD[15:0]																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 1837. PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This register must be READ ONLY when its corresponding valid bit is not asserted.

79.6.2.13 Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLWn)

Note: The reporting table is not cleared on software reset.

Figure 1983. Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLWn)

Offset: $0x0630 + n \times 0x4 + (2-1) \times 0x8$ ($n = 0$ to $25 \div 32 - 1$) ⁽¹⁾																Access: R/W			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	OFLW[31:16]																		
W	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	OFLW[15:0]																		
W	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is $94 \div 32 = 2.9375$ or $66 \div 32 = 2.0625$, round the result to 3.

Table 1838. PERIPH_RAM_OFLWn field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Section 79.7.3, Handling overflows (Multiple error reporting) , occurs.

79.6.2.14 Flash memory correctable error reporting table status register (FLASH_CERR_STS_n)

Note: The reporting table is not cleared on software reset.

Figure 1984. Flash memory correctable error reporting table status register (FLASH_CERR_STS_n)

Offset: 0x0c20 + n × 0x8 (n = 0 to 20–1)

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1839. FLASH_CERR_STS_n register field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23	Reserved
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. For non-BIST type of errors, this field is not used and reads as 0xFF Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

79.6.2.15 Flash memory correctable error reporting table address register (FLASH_CERR_ADDRn)

Figure 1985. Flash memory correctable error reporting table address register (FLASH_CERR_ADDRn)

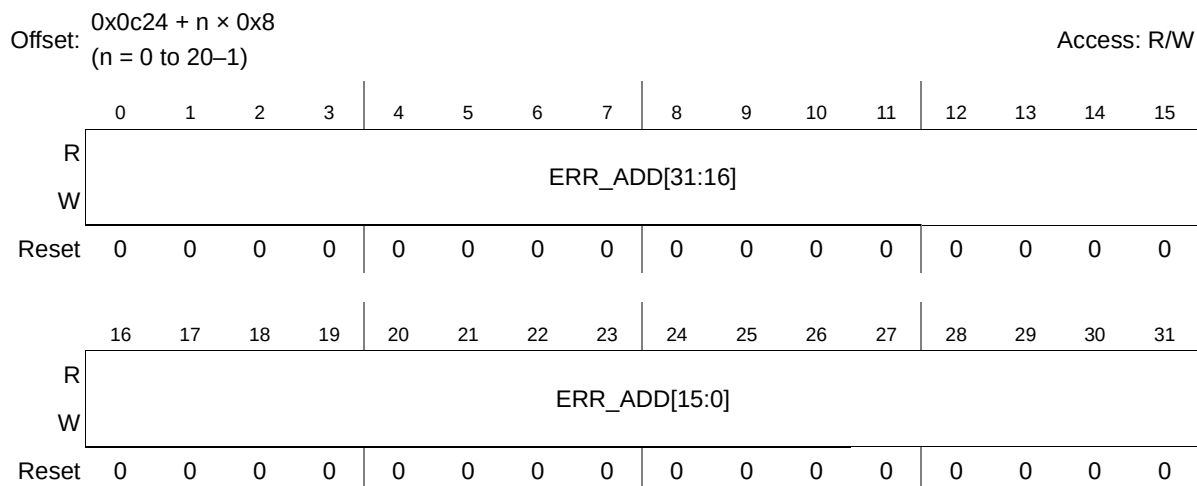


Table 1840. FLASH_CERR_ADDRn register field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

79.6.2.16 Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Figure 1986. Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

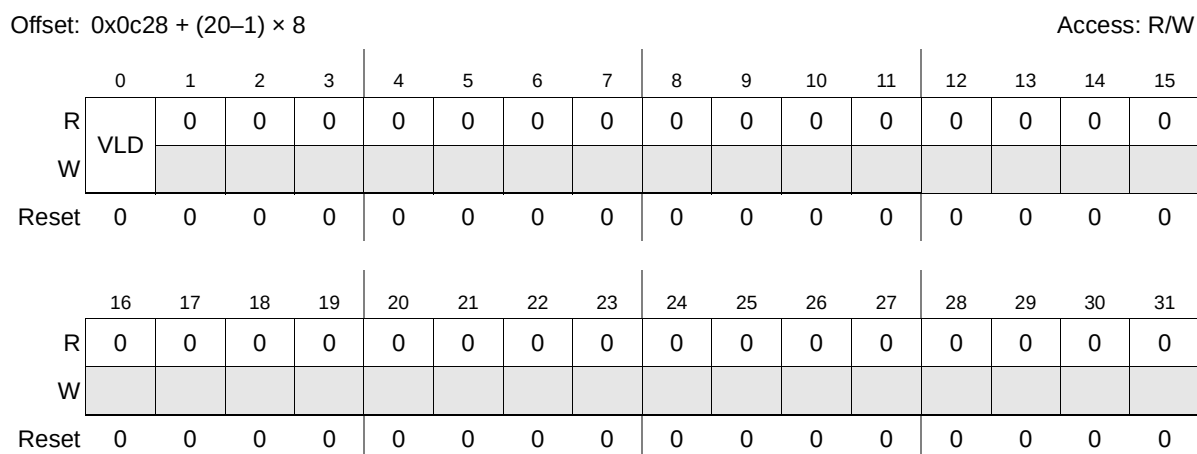


Table 1841. UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31 30–0	Reserved

79.6.2.17 Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Figure 1987. Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Offset: $0x0c2C + (20-1) \times 8$

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1842. FLASH_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This register must be READ ONLY when its corresponding valid bit is not asserted.

79.6.2.18 Flash memory concurrent overflow registers (FLASH_OFLWn)

Note: The reporting table is not cleared on software reset.

Figure 1988. Flash memory concurrent overflow registers (FLASH_OFLWn)

Offset: $0x0c30 + n \times 0x4 + (20-1) \times 0x8$
($n = 0$ to $3 \div 32 - 1$)⁽¹⁾ Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW[31:16]															
W	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW[15:0]															
W	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is $94 \div 32 = 2.9375$ or $66 \div 32 = 2.0625$, round the result to 3.

Table 1843. FLASH_OFLWn field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Section 79.7.3, Handling overflows (Multiple error reporting) , occurs.

79.7 Functional description

79.7.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known errors which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit = '1' for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address. The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows

- 1) Check that the VLD bit is not set,
- 2) Write an invalid address/bad bit information,
- 3) Set VLD bit,
- 4) Check that the address still has an invalid value,
- 5) Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any error handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 6th step, and the MEMU writes an error with the same address that Software wants to program, both the errors will be stored in the reporting table (i.e. duplicate entries).

79.7.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.

		Address [A:0]	Bad Bit [B:0]	Valid
Correctable	1			
	2			
	3			
	4			
	5			
	6			
				
				
	N			
	1			
Uncorrectable		(B+1) {1'b1}		
Flags				
Overflow Source Bits [C:0]				
		Address - The address associated with the table entry.		
		Bad Bit - (As needed) a binary encoding of the single bit that is in error.		
		Valid - A user clearable bit that indicates that the row is valid		
		Flags - Indications of important events that occurred.		
		Overflow Source Bits - A one hot encoding of the Source channel or channels that resulted in the overflow event.		

Figure 1989. Generic representation of the reporting table of one MEMU reporting block

79.7.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

- a) Error buffer overflow–More than two memories reporting an error at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
- More than 2 errors detected at same time. This would lead to overflow in the error buffer. MEMU can process only one error at a time while storing the other in the error buffer. So this would be indicated as an overflow.The uniqueness check in the input buffer does not “guarantee” that an ECC-supervised memory with only one error will not be marked as overflow source. If that one error occurs together

with two (or more) additional errors at the same cycle it can get flagged as an overflow.

- b) Correctable/Uncorrectable Error Overflow—Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
 - AN ECC error which matches the address of an MBIST error is dropped (same as if it matched the address of an ECC error). In case, where a MBIST error comes with ECC error already in the table, both the address and bad bits are compared to determine uniqueness.
- c) The Overflows will be stored in the bit-wise order i.e., if the error bit 31 reports an error, the bit 31 of the overflow register will be set to '1'.

80 Indirect Memory Access (IMA)

80.1 Introduction

Indirect Memory Access (IMA) refers to the activity of accessing any chip memory for the purpose of reading and/or modifying data, including ECC check bits. This capability is useful for test activities, e.g., verifying ECC functionality.

The SPC57EM80xx MCU provides two mechanisms for IMA:

- The built-in CPU End2End ECC test capability provides a path for accessing data and ECC check bits from some SRAM arrays
- An IMA controller module provides registers for selecting, reading, and writing memory data.

Note: Not all SRAM data and ECC bits are accessible. See the Device Configuration chapter for details.

80.2 Accessing memory via CPU End2End ECC test

The processor cores can access all bits (including ECC check bits) in SRAM arrays that support E2E ECC with the exceptions of addresses that are blocked by MPU configuration.

Note: All memory accesses by the processor cores are filtered through the MPU. There are no back doors to blocked memory addresses.

SRAM is selected by absolute memory address with check bit data in a special register. The register used depends on which memory is being accessed—I-cache/D-cache, or other memory.

Since all CPU memory accesses are arbitrated by the crossbar (XBAR), there is no need to halt normal memory access in software.

Memory is accessed via processor core registers. The register(s) used depends on the memory being accessed. This is described in the following sections.

80.2.1 Accessing I-cache and D-cache memory via processor core

For I-cache and D-cache memory, the SRAM location is selected by cache way, set and word with data and check bits in registers provided specifically for reading and writing cache tag bits and ECC check bits. There are two Device Control Registers (DCRs) provided for cache access:

- Cache Debug Access Control Register (CDACNTL)
- Cache Debug Access Data Register (CDADATA)

These registers are summarized in the following sections.

80.2.1.1 Cache Debug Access Control register (CDACNTL)

The Cache Debug Access Control Register (CDACNTL) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to I-cache and D-cache memory.

Figure 1990. Cache Debug Access Control Register (CDACNTL)

T/D	0			CWAY	0		CSET						WORD			PARITY CHECK BITS								0			CACHE	R/W	GO		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

The CDACNTL bits are described in [Table 1844](#).

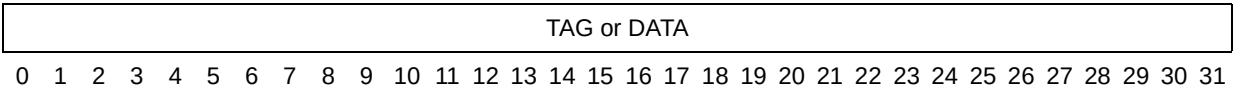
Table 1844. CDACNTL field descriptions

Bits	Name	Description
0	T/D	Tag / Data: 0 Data array selected 1 Tag array selected
1–2	—	Reserved - read as zeros
3	CWAY	Cache Way. Specifies the cache way to be selected
4–5	—	Reserved - read as zeros
6–12	CSET	Cache Set. Specifies the cache set to be selected. Only valid set numbers for the selected cache are supported, otherwise results are undefined.
13–15	WORD	Word (Used for data array access only, I or D-cache) Specifies one of eight words of selected set
16–23	PARITY CHECK BITS	Parity check bits (I or D-cache) D-cache Data array, I-cache Data Array: parity check bits for data. Tag Array: parity check bits for tag.
24–27	—	Reserved - read as zeros
28	CACHE	Cache Select. Specifies the cache to be selected. 0 Selects the data cache for the operation. 1 Selects the instruction cache for the operation.
29	R/W	Read / Write: 0 Selects write operation. Write the data in the CDADATA register to the location specified by this CDACNTL register. 1 Selects read operation. Read the cache memory location specified by this CDACNTL register and store the resulting data in the CDADATA register and store the parity bits in this CDACNTL register.
30–31	GO	GO command bits 00 Inactive or complete (no action taken) hardware sets GO=00 when an operation is complete 01 Read or write cache memory location specified by this CDACNTL register. 1x Reserved

80.2.1.2 Cache Debug Access Data Register (CDADATA)

The Cache Debug Access Data Register (CDADATA) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to I-cache and D-cache memory.

Figure 1991. Cache Debug Access Data Register (CDADATA)



The CDADATA bits are described in [Table 1845](#).

Table 1845. CDADATA field descriptions

Bits	Name	Description
0–31	TAG	TAG Array Access Data - when accessing the tag array of the I-cache: 0–19Tag compare bits 20–22Reserved 23Valid bit 24R bit 25SO bit 26–28Reserved 29–31Lockout bits. These three bits should have the same value, 1-Locked out, 0- not locked out. TAG Array Access Data - when accessing the tag array of the D-cache: 0–20Tag compare bits 21–22Reserved 23Valid bit 24R bit 25SO bit 26SW bit 27UW bit 28Reserved 29–31Lockout bits. These three bits should have the same value, 1-Locked out, 0- not locked out.
	DATA	DATA Array Access Data (Bytes 0:3 of the selected word) - when accessing the data array of either cache: 0–7 byte 0 8–15 byte 1 16–23byte 2 24–31byte 3

80.2.2 Accessing non I-cache or D-cache memory via processor core

The e2eECC Error Control/Status Register 0 (E2EECSR0) is provided specifically for reading and writing E2E ECC check bits. This register is described in the following section.

80.2.2.1 e2eECC Error Control/Status Register 0 (E2EECSR0)

The e2eECC Error Control/Status Register 0 (E2EECSR0) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to E2E ECC data. It is shown in the following figure.

Figure 1992. e2eECC Error Control/Status Register 0 (E2EECSR0)

0				RCHKBIT								0				WRC	INVC	0				WCHKBIT/CHKINVT									
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

The E2EECSR0 bits are described in [Table 1846](#).

Table 1846. CDACNTL field descriptions

Bits	Name	Description
0–3	—	Reserved - read as zeros
4–11	RCHKBIT	Read Checkbits. This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the data BIU. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation, assuming interrupts are masked.
12–15	—	Reserved - read as zeros
16–17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) with the values in the WCHKBIT control field. Software must clear this field before re-writing to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10, by driving the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved <i>Note: Checkbit substitution has priority over Error injection</i>
18–19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying p_hwchkbit[7:0] based on CHKINVT. Software must clear this field before re-writing to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying p_hwchkbit[7:0] based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved <i>Note: Checkbit substitution has priority over Error injection</i>
20–23	—	Reserved - read as zeros
24–31	WCHKBIT/CHKINVT	Write Checkbits / Checkbit Invert Mask. This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out when error injection occurs.

80.3 Accessing memory via the IMA module

The Indirect Memory Access (IMA) module provides an alternative way to access on-chip memory without going through standard peripheral or system RAM interfaces to check the data or change the data.

This alternative path can be used for diagnostic purposes. For example, for SRAM arrays that do not support E2E ECC, the IMA block provides a way to write corrupted ECC values so that the ECC error decoding logic can be checked.

Please note the following restrictions on memory access via IMA:

- ECC must be disabled prior to IMA write access. If ECC is enabled and a write access is attempted to ECC-bits, the write access is blocked and an error signal is forwarded to the FCCU and the corresponding error flag is set. Read access is always allowed.
- The IMA must be unlocked for reads

The order of precedence for access to a memory is shown below. Once the IMA has been granted access, no other accesses are allowed until the IMA select line to the selected memory is negated by the IMA.

Note: Once the IMA has selected a RAM, it expects no other accesses from any other master until deselected by the IMA.

The IMA module signals the FCCU when it is reading or writing to a memory address. This enables the FCCU to assess whether the access is intentional or spurious. For an intentional access the respective FCCU error input needs to be disabled, otherwise it should be enabled.

80.3.1 Features

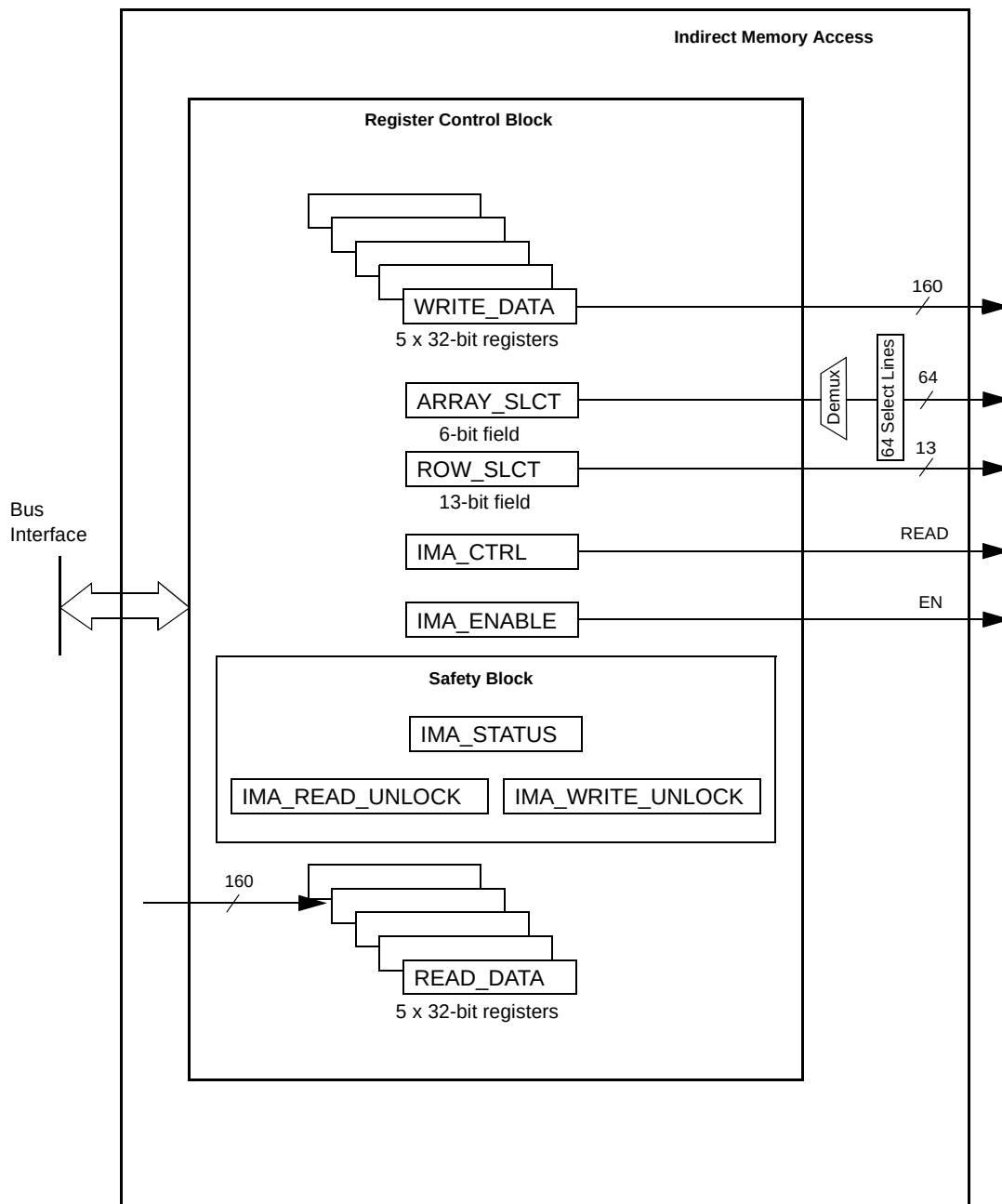
The IMA module includes the following features:

- Able to read all bits in all RAM arrays connected to the IMA, once the IMA is unlocked for reads
- Enables reading and manipulation of ECC check bits.
- During an IMA SRAM access, other accesses are affected as follows:
 - Reads return random data.
 - Writes have no effect.
- Any IMA access to memory are indicated in the Fault Collection and Control Unit (FCCU).
- IMA accesses to memory do not bypass any SRAM repair which may have occurred during production test of the device.
- IMA accesses to memory use a physical address, not a logical address, and thus memory addresses used by the IMA are not memory mapped.

80.3.2 Block diagram

The following figure shows the structure and function of the IMA module. The Register Control Block contains the memory-mapped registers accessible by software. The Safety Block prevents spurious activation of the IMA module. It ensures the IMA does not access RAM unless the system needs it to.

Figure 1993. IMA block diagram



80.3.3 IMA Module Memory Map and Registers

This section provides details of the IMA memory map. [end] It also provides detailed descriptions for each of the registers in that map.

The Indirect Memory Access unit's memory map allows for the CPU to access RAM's on the chip to write test data so that single-bit ECCs and multi-bit ECC logic can be tested.

Table 1847. IMA Memory Map

Offset or Address	Register	Access	Reset Value	Section/Page
0000_0000	IMA Control Register (IMA_CTRL)	R/W ⁽¹⁾	0x0000_0000	80.3.3.1/3464
0000_0004	IMA Enable Access Register (IMA_ENABLE)	R/W	0x0000_0000	80.3.3.2/3465
0000_0008	IMA Status Register (IMA_STATUS)	R	0x0000_0101	80.3.3.3/3466
0000_000C	IMA RAM SELECT Register (IMA_SLCT)	R/W	0x0000_0000	80.3.3.4/3467
0000_0010	IMA Write Unlock Register (IMA_WRITE_UNLOCK)	R/W	0x0000_0000	80.3.3.5/3468
0000_0014	IMA Read Unlock Register (IMA_READ_UNLOCK)	R/W	0x0000_0000	80.3.3.6/3469
0000_002C	IMA RAM Write Data Register 4 (IMA_WRITE_DATA_4)	R/W	0x0000_0000	80.3.3.7/3470
0000_0030	IMA RAM Write Data Register 3 (IMA_WRITE_DATA_3)	R/W	0x0000_0000	80.3.3.7/3470
0000_0034	IMA RAM Write Data Register 2 (IMA_WRITE_DATA_2)	R/W	0x0000_0000	80.3.3.7/3470
0000_0038	IMA RAM Write Data Register 1 (IMA_WRITE_DATA_1)	R/W	0x0000_0000	80.3.3.7/3470
0000_003C	IMA RAM Write Data Register 0 (IMA_WRITE_DATA_0)	R/W	0x0000_0000	80.3.3.7/3470
0000_004C	IMA RAM Read Data Register 4 (IMA_READ_DATA_4)	R	0x0000_0000	80.3.3.8/3471
0000_0050	IMA RAM Read Data Register 3 (IMA_READ_DATA_3)	R	0x0000_0000	80.3.3.8/3471
0000_0054	IMA RAM Read Data Register 2 (IMA_READ_DATA_2)	R	0x0000_0000	80.3.3.8/3471
0000_0058	IMA RAM Read Data Register 1 (IMA_READ_DATA_1)	R	0x0000_0000	80.3.3.8/3471
0000_005C	IMA RAM Read Data Register 0 (IMA_READ_DATA_0)	R	0x0000_0000	80.3.3.8/3471

1. Note that R/W registers may contain some read-only or write-only bits.

80.3.3.1 IMA Control Register (IMA_CTRL)

The IMA CTRL register controls read/write accesses. Since the IMA may interfere with proper read/write access of RAMs, a locking scheme is implemented to prevent accidental activation of the IMA. Using this scheme, the IMA read/write accesses can be locked when desired.

Figure 1994. IMA Control Register (IMA_CTRL)

Offset 0x00												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1848. IMA_CTRL field descriptions

Field	Description
0–30 Reserved	
31 READ	IMA Read. This bit controls whether the IMA will perform a read or write 0 The IMA access will be a write. 1 The IMA access will be a read.

80.3.3.2 IMA Enable Access Register (IMA_ENABLE)

The Enable Access Register is a single-bit register that starts an IMA access. The bit automatically clears itself after two clocks thus providing an enable pulse.

Note: *Back to back writes to the enable bit is not valid. It will still generate a two clock enable pulse and will be considered as a single access initiation.*

Figure 1995. IMA Enable Access Register (IMA_ENABLE)

Offset 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1849. IMA_ENABLE field descriptions

Field	Description
0–30 Reserved	
31 EN	IMA Enable. This bit controls RAM access initiation by the IMA. Once set, this bit is cleared after two clocks to provide an enable pulse. 0 No IMA access will occur or the IMA has finished the previous access. 1 Start an IMA access or an IMA access is occurring.

80.3.3.3 IMA Status Register (IMA_STATUS)

The IMA Status Register shows the current status of the IMA as to which operations are unlocked.

Figure 1996. IMA Status Register (IMA_STATUS)

Offset 0x08								Access: User read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	WRITE_LOCK	0	0	0	0	0	0	0	READ_LOCK
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Table 1850. IMA_STATUS field descriptions

Field	Description
0–22 Reserved	
23 WRITE_LOCK	The ability of the IMA to do write accesses. The WRITE_LOCK bit shows the current status of the lock on IMA write accesses. If set, the IMA does not write to the memories. If cleared, the IMA is allowed to do writes to the memories. 0 Write accesses from the IMA to memories are allowed 1 Write accesses from the IMA to memories are not allowed. Any memory writes from the IMA are ignored.
24–30 Reserved	
31 READ_LOCK	The ability of the IMA to do read accesses. The READ_LOCK bit shows the current status of the lock on IMA read accesses. If set, the IMA does not do reads from the memories. If cleared, the IMA is allowed to do reads from the memories. 0 Read accesses from the IMA to memories are allowed. 1 Read accesses from the IMA to memories are not allowed. Any memory reads from the IMA are ignored.

80.3.3.4 IMA RAM Select Register (IMA_SLCT)

The IMA RAM Select register provides a method of selecting which RAM memory block will be accessed. This method is used because RAM is to be accessed in a non-functional way, i.e., accessing all bits including the ECC, and accessing on RAM word boundaries instead of byte or 32-bit word boundary.

Note: Only one RAM can be selected at a time. Each SoC has a different encoding for the RAMs. See the Device Configuration chapter for details.

Figure 1997. IMA RAM SELECT Register (IMA_SLCT)

Offset 0x0C												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	ROW_SLCT												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ARRAY_SLCT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1851. IMA_SLCT field descriptions

Field	Description
0–2 Reserved	
2–15 ROW_SLCT	The ROW_SLCT value specifies the row of the current RAM array under test to be selected. Hence the largest RAM which the IMA can access is an 8 KB word RAM.
16–25 Reserved	
26–31 ARRAY_SLCT	The ARRAY_SLCT specifies the RAM array to be tested. Up to 64 RAM instances can be supported.

80.3.3.5 IMA Write Unlock Register (IMA_WRITE_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The WRITE_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for writes.

Figure 1998. IMA Write Unlock Register (IMA_WRITE_UNLOCK)

Offset 0x10				Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	WRITE_KEY																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	WRITE_KEY																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1852. IMA_WRITE_UNLOCK field descriptions

Field	Description
0–31 WRITE_KEY	Key to unlock the IMA. – 0x04A43F95 is the first key that has to be written to start the unlock process. – 0xE4A9EBF7 is the second key that has to be written to unlock the IMA.

80.3.3.6 IMA Read Unlock Register (IMA_READ_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The READ_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for reads.

Figure 1999. IMA Read Unlock Register (IMA_READ_UNLOCK)

Offset 0x14				Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	READ_KEY																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	READ_KEY																		
W																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1853. IMA_READ_UNLOCK field descriptions

Field	Description
0–31 READ_KEY	Key to unlock the IMA Read Access to RAMs. – 0xF06AB5BC is the first key that has to be written to start the unlock process. – 0x14081B56 is the second key that has to be written to unlock the IMA.

80.3.3.7 IMA RAM Write Data Register n (IMA_WRITE_DATA_n)

The IMA_WRITE_DATA_n registers hold the data values to be written to the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a write access.

Figure 2000. IMA RAM Write Data Register n (IMA_WRITE_DATA_n)

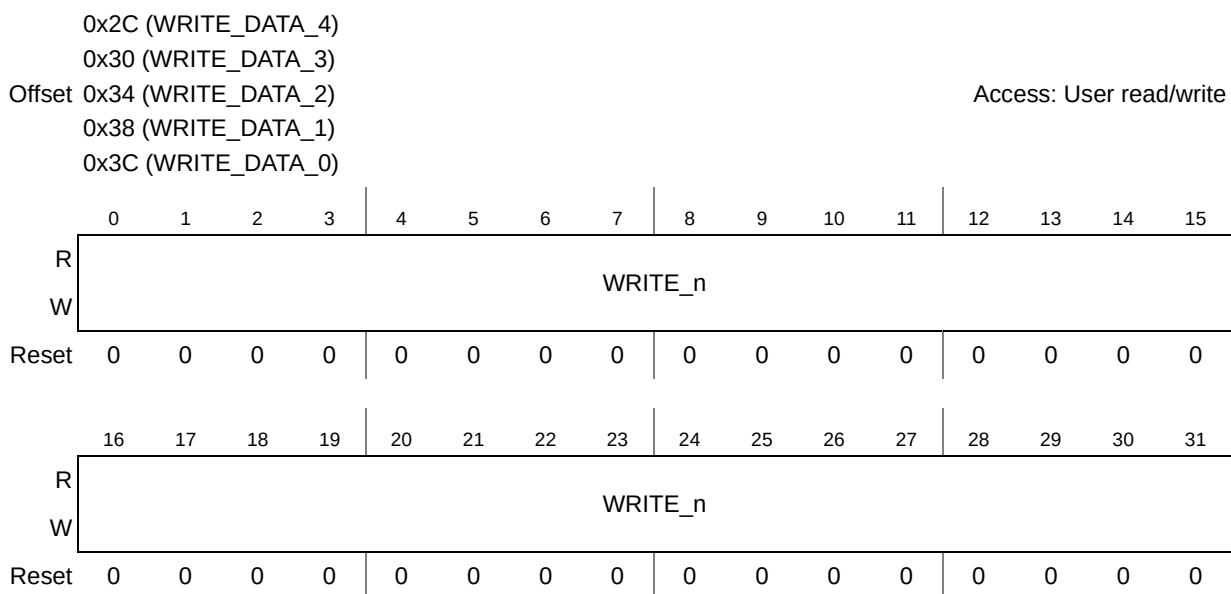


Table 1854. IMA_WRITE_DATA_n field descriptions

Field	Description
0–31 WRITEn	Contains data to be written to RAM. – WRITE_DATA_4[WRITE_4] contains write data bits[159:128] – WRITE_DATA_3[WRITE_3] contains write data bits[127:96] – WRITE_DATA_2[WRITE_2] contains write data bits[95:64] – WRITE_DATA_1[WRITE_1] contains write data bits[32:63] – WRITE_DATA_0[WRITE_0] contains write data bits[31:0]

80.3.3.8 IMA RAM Read Data Register n (IMA_READ_DATA_n)

The IMA_READ_DATA_n registers hold the data values read from the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a read access. Performing a read operation on IMA_READ_DATA_n registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Figure 2001. IMA READ_DATA_n Register (IMA_READ_DATA_n)

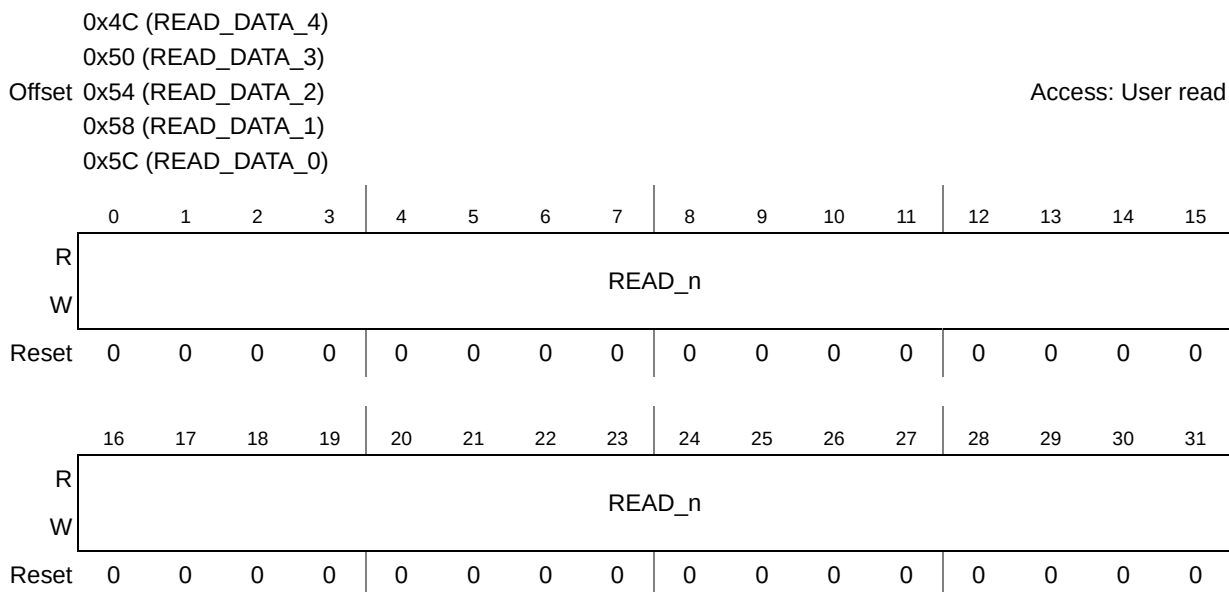


Table 1855. IMA_READ_DATA_n field descriptions

Field	Description
0–31 WRITEn	Contains data to be written to RAM. – READ_DATA_4[READ_4] contains read data bits[159:128] – READ_DATA_3[READ_3] contains read data bits[127:96] – READ_DATA_2[READ_2] contains read data bits[95:64] – READ_DATA_1[READ_1] contains read data bits[32:63] – READ_DATA_0[READ_0] contains read data bits[31:0]

80.3.4 IMA module functional description

This section describes the operation of the IMA, beginning with the unlocking of reads and writes, the software initialization, then the software interface for generating single-bit ECC errors, generating multiple-bit ECC errors, and a software supported memory BIST.

80.3.4.1 Initialization sequence

This section describes which registers are reset due to hardware reset and what locations the user must initialize prior to doing an access by the IMA.

80.3.4.2 Hardware controlled initialization

In the IMA, registers and control logic are reset by hardware (system reset). A system reset deasserts output signals and resets general configuration bits.

80.3.4.3 User initialization (prior to asserting IMA_ENABLE[EN])

The user needs to initialize portions of the SoC prior to setting the IMA_ENABLE[EN] bit. The exact values depend on the particular application. Please consult the specific chapter's memory map to see how to turn off that peripheral so that accesses are not happening at the same time as the IMA. Without the peripheral shut down, there is a small chance that the IMA and the peripheral can interfere with each other in accessing the RAM.

80.3.4.3.1 Shut down the peripheral

The IMA can cause problems with the system or the peripheral if both the IMA and the other system components are accessing the RAM during normal operation.

To shut down the peripheral, please read the peripheral's documentation. The documentation will describe how to put the peripheral into a quiet state so that it will not access the RAM during the IMA access. If the other functional path is running at the same time, the data for the other peripheral will be corrupted and unknown.

80.3.4.3.2 Unlocking reads

The IMA will not allow a read to any memory unless it has been unlocked. This is a safety feature so that it will take two unique writes to the READ_KEY register to unlock writes. To unlock the read feature, the system will have to do the following:

1. Write 0xF06AB5BC to READ_KEY
2. Write 0x14081B56 to READ_KEY

The IMA has two features to make the READ_KEY lock in the following instance:

- Writing an incorrect key to READ_KEY

80.3.4.3.3 Unlocking writes

(The IMA will not allow a write to any memory unless it has been unlocked. This is a safety feature so that it will take two unique writes to the WRITE_KEY register to unlock writes. To unlock the write feature, the system will have to do the following:

1. Write 0x04A43F95 to WRITE_KEY
2. Write 0xE4A9EBF7 to WRITE_KEY

The IMA has a feature to make the WRITE_KEY lock in the following two instances:

1. Writing an incorrect key to WRITE_KEY
2. Not setting the IMA_ENABLE[bit] for 16,384 system clocks.

80.3.4.4 IMA lock

The IMA module also allows for read/write access locking via an input pin. If this input is asserted, both read/write accesses via IMA are locked. This locking scheme overrides the above KEY based locking scheme. Hence it provides additional safety over the KEY based locking method. However, usage of this feature is entirely SOC specific.

80.3.4.5 IMA access

There are two differences between a read access and a write access. The first difference is that the IMA_STATUS[WRITE_LOCK] bit has to be cleared for writes and the IMA_STATUS[READ_LOCK] has to be cleared for reads. The second difference is that the IMA_CTRL[READ] has to be set for reads and IMA_CTRL[WRITE] has to be cleared for writes.

The following sections show the different accesses that are available depending on the options.

80.3.4.5.1 Read access

Read Accesses are expected to follow the sequence below

1. Unlock Read Access by performing appropriate consecutive writes to READ_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Set the READ bit in the IMA_CTRL register
4. Set the EN bit in the IMA_ENABLE register to initiate read access. The EN bit will self-clear after two clocks.
5. Software must wait an appropriate number of clocks depending on the speed of the SRAM for the access to complete.
6. Read data from READ_DATA registers. This causes data from RAM to be latched into the IMA and returned to the CPU on the next clock.

80.3.4.5.2 Write access

Write Accesses are expected to follow the sequence below

1. Unlock Read Access by performing appropriate consecutive writes to READ_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Write the Write Data in WRITE_DATA registers.
4. Clear the READ bit in the IMA_CTRL register
5. Set the EN bit in the IMA_ENABLE register to initiate write access. The EN bit will self-clear after two clocks.
6. Software must wait an appropriate number of clocks for the IMA write to occur, depending on the speed of the SRAM being written.

80.3.5 Application information

Warning: The IMA will inhibit correct operation of the underlying peripherals. The application software is responsible for ensuring there is not a resource conflict.

The primary function of the IMA module is to provide a means for the system to program invalid ECC bits and see that the appropriate peripheral is able to detect the error. For a

single-bit error, the peripheral should be able to correct the error. For a multiple-bit error, the peripheral should report an error.

81 Fault Collection and Control Unit (FCCU)

81.1 Introduction

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect errors and to place the device into a safety state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

The FCCU offers a systematic approach to fault detection and control. The distinctive features of the module are:

- Redundant collection of hardware checker (e.g. RCCU) results
- Redundant collection of error information from safety relevant modules on the device
- Collection of test results
- Configurable and graded fault control
- Internal reactions (independently configurable for each RF):
 - No reset reaction
 - IRQ
 - Short functional reset
 - Long functional reset
 - NMI
- External reaction (failure is reported to the outside world via one or more output pins)
- Watchdog timer for the reconfiguration phase
- The FCCU's configuration is lockable in two ways:
 - Permanently locked until next reset or transiently locked until a specific key is written. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (ie., other than 0XBC). Key to lock the FCCU permanently is 0XFF, written in the FCCU_TRANS_LOCK register.
- One of the failure indication pins is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the failure indication protocol is configured to be a toggling protocol i.e. dual rail and time switching.
- After power-on the error out pins have high impedance. FCCU goes to operational state only on software request.
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_{min} , even if SW tries to release it before (for the case of error pin configured in Bi-stable mode only). $T_{min} = 250 \mu s + \Delta T$, with ΔT parameter being configurable by SW up to 10 ms.
- The overall error detection, processing and indication time is less than 10 ms.
- The actual maximum time is 5 safe (IRCOSC) clock cycles.

Two classes of fault are identified based on the criticality and the related reactions. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable based on the fault criticality. The default configuration can be modified only in a specific FCCU state for application/test/debugging purposes.

81.1.1 Acronyms and abbreviations

Table 1856. Acronyms and abbreviations

Term	Description
UF	Unrecoverable fault
CPU	Central processing unit
EOUT	Error out
FOSU	FCCU output supervision unit
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRQ	Interrupt request
MC	MC_CGM, MC_ME, MC_PCU, and MC_RGM modules
RF	Recoverable fault
NMI	Non-maskable interrupts
NVM	Nonvolatile memory
RCC	Redundancy control checkers
RCCU	Redundancy control checker unit
SoC	System-on-chip
STCU	Self-test control unit
SW	Software
WKPU	Wake-up unit

81.2 Main features

- 1 to 128 recoverable faults management
- HW or SW fault recovery management
- Fault detection collection
- Fault injection (fake faults; see the implementation-specific details)
- External reaction (fault state): EOUT signaling. Failure indication via the pin(s) is controlled by the FCCU.
- Internal (SoC) reactions (alarm state): interrupt request
- Internal (SoC) reactions (fault state): long functional reset request pulse, short functional reset request pulse, NMI
- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Watchdog timer for the reconfiguration phase
- Configuration lock

81.2.1 Standard features

- IPS (slave bus signals) bus interface

81.3 Block diagram

The top-level diagram of the FCCU module is given in [Figure 2002](#).

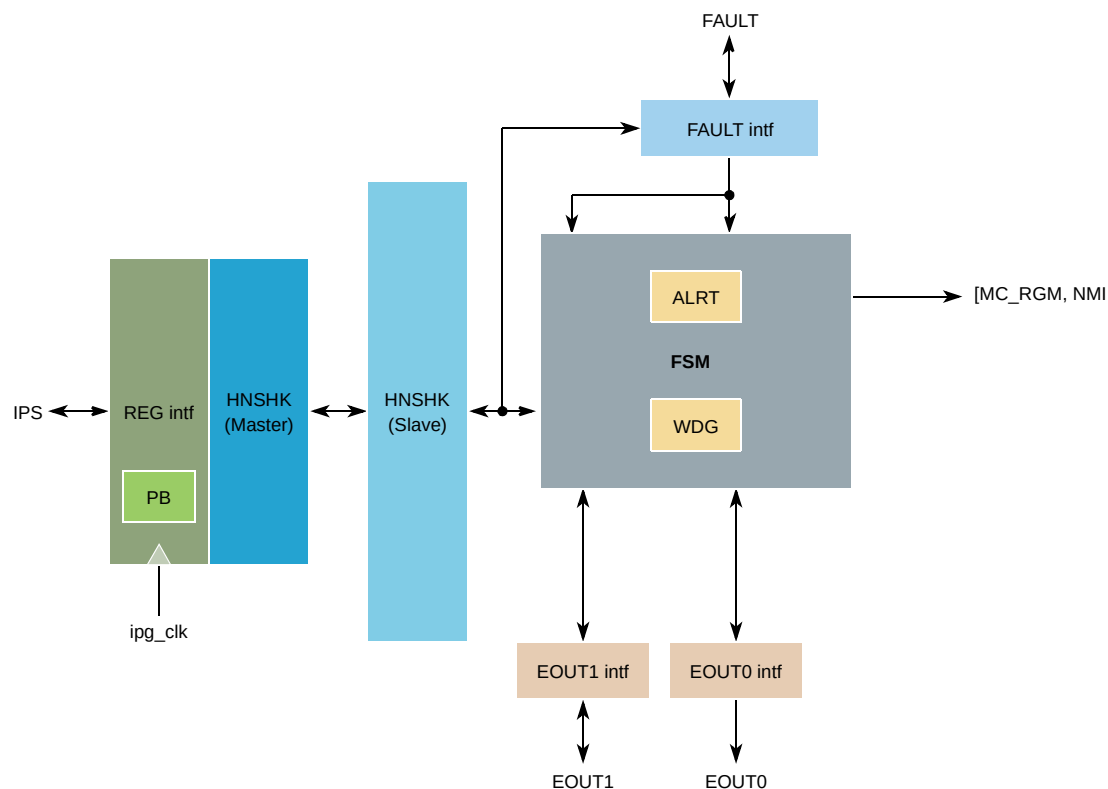


Figure 2002. FCCU block diagram

The FCCU module includes the submodules listed in [Table 1857 \(FCCU submodules\)](#).

Table 1857. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSMs to support the handshake between the REG if and the FSM unit due to the usage of 2 asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: – Watchdog timer (WDG) – Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

81.4 Signal description

81.4.1 Pinout

The FCCU generates two external signals, EOUT[0] and EOUT[1]. These are described in [Section 81.5.9, EOUT interface](#).

For the availability of these signals on a chip, see the device configuration chapter.

81.4.2 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported:

- word (32 bits) data write/read operations to any registers
- low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data write/read operations to any registers
- any other operation (free byte enables or other operations) must be avoided.

The FCCU module generates a transfer error in the following cases:

- any write/read access executed outside the address space of the peripheral.
- any write/read operation different from byte/hword/word (free byte enables or other operations) on each register.
- any write access to the configuration registers not executed in CONFIG state
- any write access to the Transient and permanent Lock Register in any mode other than supervisor access mode

The registers of the FCCU module are accessible (read/write) in each access mode: user, supervisor or test.

81.5 Functional description

81.5.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is asserted. That is, if 0 on the fault signal indicates fault, then the status flags are valid until the fault line is '0'. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched in an external module at the FCCU. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the error is recoverable by itself when the error condition gets corrected.

The following types of reset are applicable:

- Destructive reset: any type of reset related to a power failure condition that implies a complete system re-initialization
- Long functional reset: implies the Flash and digital circuitry (most of it except FCCU, STCU) initialization
- Short functional reset: implies the digital circuitry (most of it except FCCU, STCU) initialization.

81.5.2 FSM description

The FCCU module functionality is depicted by the FSM diagram given in [Figure 2003](#).

Basically four states are identified with the following meaning:

- **CONFIG:** The configuration state is used only to modify the default configuration of the FCCU. A sub-set of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, time-out, recoverable fault masking) can be accessed in write mode only in the CONFIG state.
The CONFIG state is accessible only in NORMAL state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset of the FCCU, the transient lock register is unlocked by writing BCh into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (ie., other than 0XBC). Key to lock the FCCU permanently is 0XFF, written in the FCCU_TRANS_LOCK register. After reset lifts, the state shall be transiently locked (permanent lock → is unlocked and transient lock → is locked).
The CONFIG to NORMAL state transition can be executed by SW or automatically following a time-out condition of the watchdog. In case the timeout information and the SW request to mode change to NORMAL appears at the same time, watchdog timeout had the priority and hence the Configuration registers (those that are writeable only in CONFIG mode) are reset to their default values. The movement to NORMAL is made. The incoming faults, occurring during the configuration phase (CONFIG state) are anyway latched in order to process them when the FCCU is moved into the NORMAL state, according to the new configuration.
- **NORMAL:** This is the FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following one of the following events:
 - unmasked recoverable faults with the time-out disabled → the FCCU moves to the FAULT state
 - unmasked recoverable faults with the time-out enabled → the FCCU moves to the ALARM state
 - masked recoverable faults → the FCCU stays in NORMAL state
- **ALARM:** the FCCU moves into the ALARM state when a unmasked recoverable fault occurs and the time-out is enabled. The transition to the ALARM state goes along with an interrupt (irq_alarm), if enabled. By definition, this fault may be recovered within a programmable time-out period, before it generates a transition to the FAULT state. The time-out is re-initialized if the FCCU state moves to the NORMAL state. The time-out restarts following the recovery from the FAULT state.
- **FAULT:** the FCCU moves into the FAULT state when one of the following condition occurs:
 - time-out related to a recoverable fault when the FCCU is in the ALARM state
 - unmasked recoverable faults with the time-out disabled

The transition from NORMAL/ALARM state goes along with the generation of:

- NMI interrupt (optional)
- EOUT signaling (optional)
- SW option: Soft reaction (Short functional reset request pulse if configured)
- SW option: Hard reaction (Long functional reset request pulse if configured)

Non Maskable Interrupt (NMI) is routed only to the Safety Core.

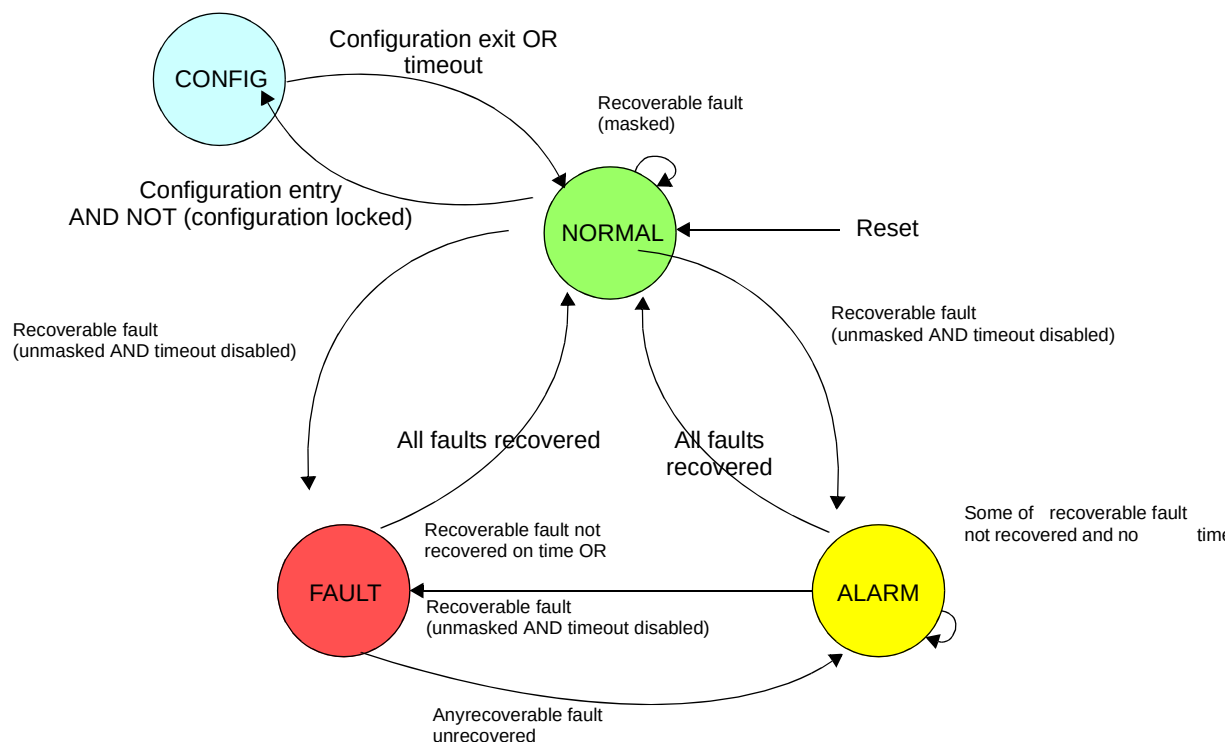


Figure 2003. FCCU state diagram

81.5.3 Reset interface

The FCCU has two input resets, as described in [Table 1858](#) and two output resets related to the FAULT state. In FAULT state, each fault can be programmed to generate a soft or hard reaction request pulse.

In [Table 1859](#) for each type of fault reaction (hard, soft or self-checking), the respective output reset (rst_lfunc_b, rst_sfunc_b) is activated. In [Table 1860](#) for each type of reset source (external reset, POR, STCU, FCCU failure, short functional, long functional), the respective FCCU input reset state is described.

The MC unit is responsible to generate the corresponding input reset as described in [Table 1859](#) and [Table 1860](#). [Table 1859](#) and [Table 1860](#) describe a typical reset strategy at system level. [Table 1861](#) summarizes the MC actions corresponding to each reset generated by FCCU.

Table 1858. Input reset

Input reset	Description
ipg_hard_asynch_rst_clk_b	Global reset of the FCCU (synchronous to the system clock)
ipg_hard_asynch_rst_clk_safe_b	Global reset of the FCCU (synchronous to the IRCOSC)

Table 1859. Reset reactions

FAULT reaction	Active reset	ipg_hard_asynch_rst_*_b
Hard	rst_lfunc_b	disabled
Soft	rst_sfunc_b	disabled

[Table 1860](#) shows the Reset sources.

Table 1860. Reset sources

Reset source	ipg_hard_asynch_rst_*_b
External reset	enabled
POR	enabled
STCU	enabled
FCCU failure	application dependent
Functional reset	disabled

Table 1861. MC actions

Active reset	MC action
rst_lfunc_b	Long functional reset (FLASH + digital modules excluded FCCU, STCU)
rst_sfunc_b	Short functional reset (digital modules excluded FCCU, STCU)

In reset state the FCCU is not able to collect any RF. The FCCU reset phase (POR, external reset, STCU, destructive reset, FCCU failure) must guarantee the assertion of the FCCU input resets at least for 2 IRCOSC clock periods.

81.5.4 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (and recoverable) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a recoverable fault (unmasked and with time-out disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the recoverable faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the recoverable faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been

cleared (SW or HW recovery) otherwise the FCCU will return in the ALARM state (if any recoverable fault is still pending and the time-out is not elapsed).

In general, no fault nesting is supported except for the recoverable faults that causes a ALARM to FAULT state transition. In this case the NCT timer is stopped until the FAULT state is recovered.

81.5.5 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a recoverable FAULT management (ALARM state), see [Figure 2004](#) and [Figure 2005](#), is as follows:

- recoverable fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
 - alarm interrupt request (if configured)
 - time-out running
- system state: RUN
- alarm interrupt management: FAULT recovery (by SW)—FCCU state transition ALARM → NORMAL

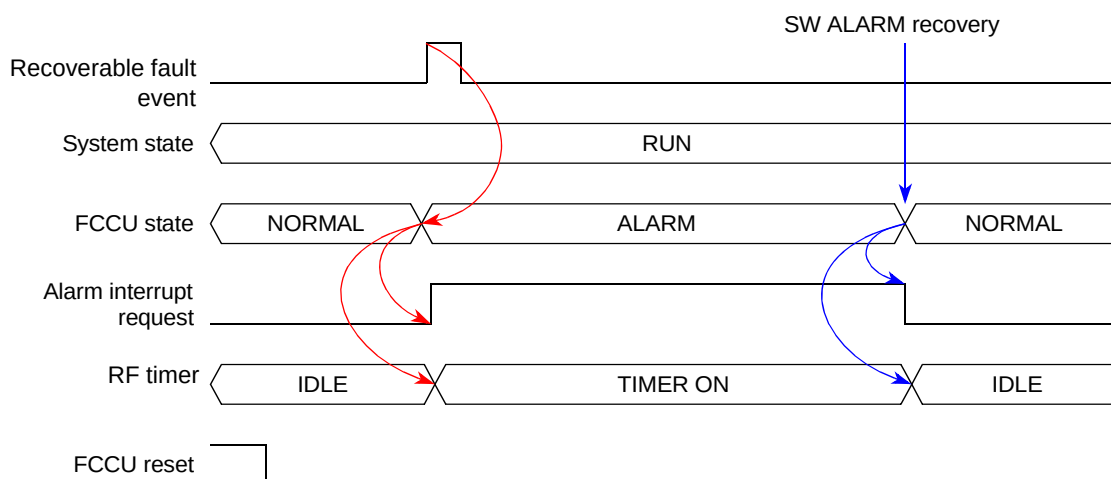


Figure 2004. Recoverable FAULT (ALARM state) recovery (a)

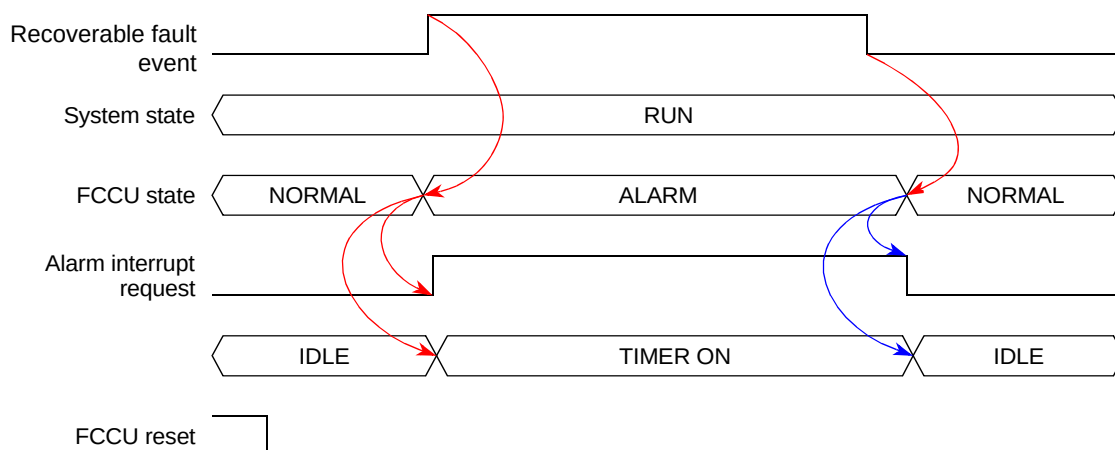


Figure 2005. Recoverable FAULT (ALARM state) recovery (b)

A typical sequence related to a recoverable FAULT management (ALARM → FAULT state), see [Figure 2006](#), is as follows:

- recoverable fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
 - alarm interrupt request (if enabled)
 - time-out running
- FCCU state transition (following the time-out trigger): ALARM → FAULT — xNMI assertion (if enabled)
- system state transition: RUN → SAFE
- NMI interrupt management (if enabled)
 - FAULT recovery (by SW): FCCU state transition FAULT → NORMAL
 - system state transition: SAFE → RUN

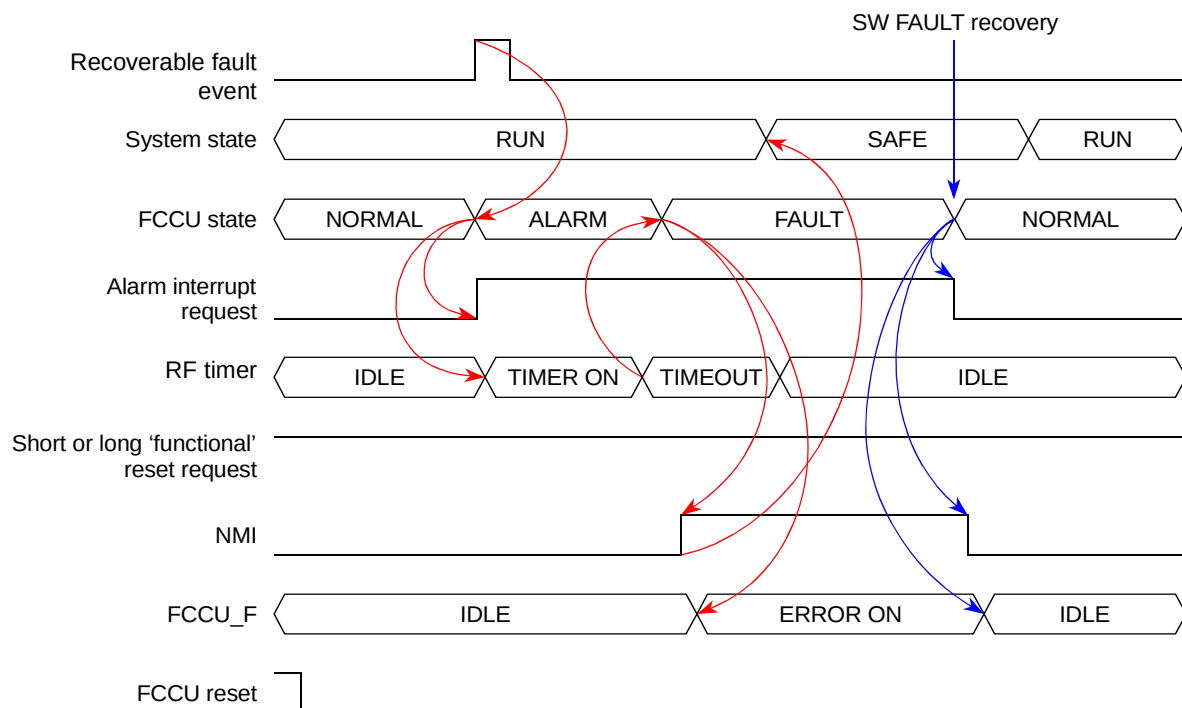


Figure 2006. Recoverable FAULT (ALARM -> FAULT state) recovery

81.5.6 NMI interface

BLG The NMI signal internally generated by FCCU is masked when:

- FCCU asynchronous reset
- MC's current mode (`mc_state[3:0]`) = RESET

and unmasked when a SW change request of the MC state is triggered (`sw_mode_change_req` = 1).

81.5.7 FAULT interface

The basic functionality of the fault interface is to provide a simple programmable edge sensitive interface (active high or low) to signal a fault condition to the FCCU module.

To support a fault injection mechanism, in order to generate fake faults directly in the FAULT root, an additional and optional signal is available. It is an active high signal and is asserted for a single clock cycle synchronously at the system clock. The behavior is given in [Figure 2007](#). The fault injection is executed by a write operation in the FCCU_RFF register. The reaction following a fake recoverable fault can be masked. The fault injection mechanism is optional; see the implementation-specific details.

To support a fault clearing mechanism, in order to clear a fault latched directly in the FAULT root, an additional and optional signal (`fccu_rf_clr`) is available. This active-high signal is asserted for a single clock cycle synchronously at the system clock. The behavior is given in [Figure 2008](#). The fault clearing is executed when the SW "clears" the status (flag) bits of the

FCCU_RF_Sx. The fault clearing mechanism is optional and its support must be specified at system level.

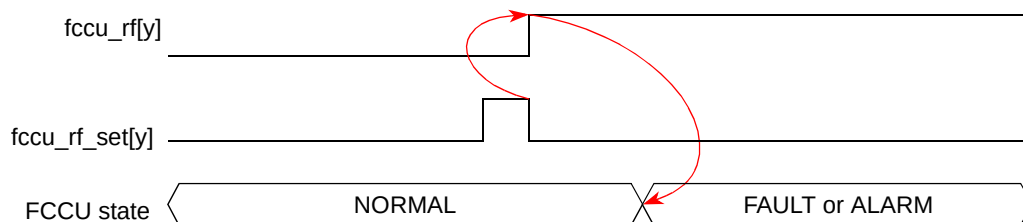


Figure 2007. FAULT injection

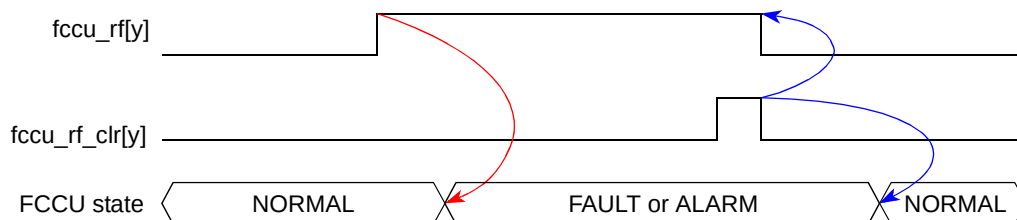


Figure 2008. FAULT clearing

81.5.8 STCU interface

The STCU interface includes:

- a set of signals resulting from the self-checking procedure connected externally at these FCCU faults:
 - recoverable

The STCU fault signals are processed by the FCCU when the SOC is rebooted following the self-testing procedure. The STCU includes also a status register that stores the self-testing results (flags).
- a mask (`fccu_eout_msk[1:0]`) that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self testing procedure, depending on the STCU results, three cases are applicable:

- STCU completes the self testing procedure successfully. The SOC reboots and the FCCU is responsible to provide a reaction.
- STCU completes the self testing procedure with low severity failures. The FCCU is responsible to provide the proper reactions according to the fault occurred.
- STCU completes the self testing procedure with serious failures. The FCCU or other critical parts of the SoC could not be able to provide the proper reaction. STCU should keep permanently the SoC in RESET state. This feature is optionally programmable inside the STCU.

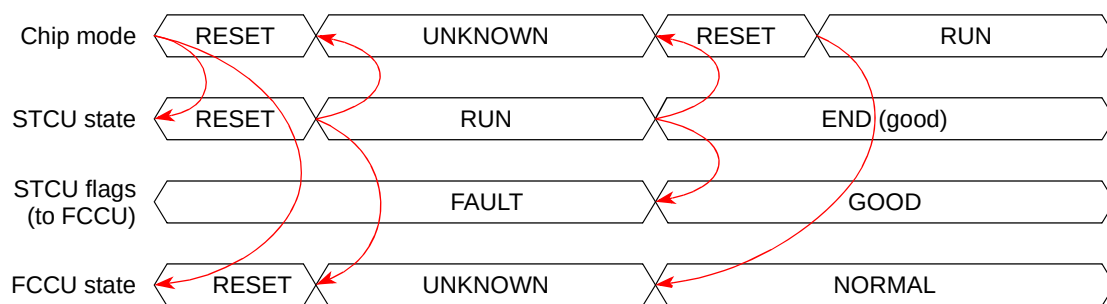


Figure 2009. STCU-FCCU (case a)

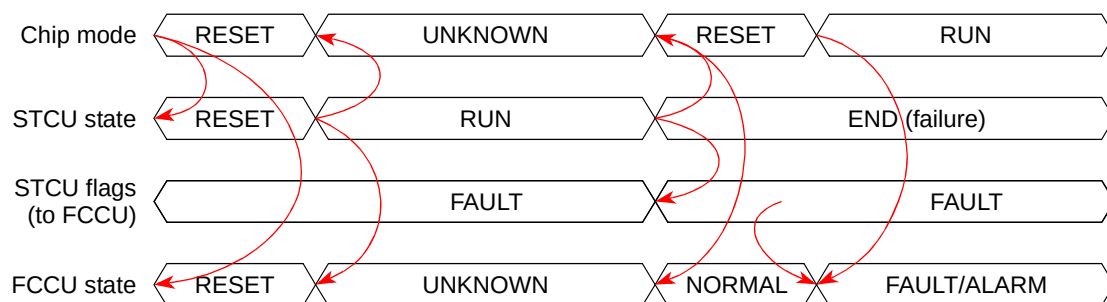


Figure 2010. STCU-FCCU (case b)

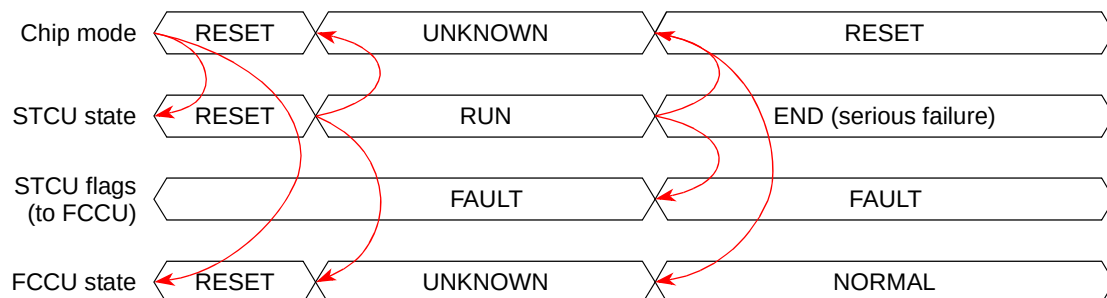


Figure 2011. STCU-FCCU (case c)

81.5.9 EOUT interface

The FCCU provides up to two bidirectional signals (EOUT interface) as a failure indication to the external world (see table “FCCU failure inputs” in the “Functional safety” chapter for details). These signals need to be mapped to appropriate pins for external visibility. Since SIUL2 does not control the pad buffer enable during Alternate function, FCCU provides a mechanism to drive the pad in OD (Open Drain) mode where the pad is driven Hi-Z only when FCCU drives logic ‘1’ else it drives logic ‘0’ (when FCCU drives logic ‘0’). The selection

to OD is done by OD bit in FCCU_CFG register. Different protocols for the EOUT interface are supported, selecting the FCCU_CFG.FOM register field:

- Dual rail protocol
- Time switching protocol
- Bi-stable protocol
- Test mode

The signal polarity can be programmed by setting the PS fields in the FCCU_CFG register. For example, for PS = 0, fcc_eout[0] = 0 and fcc_eout[1] = 1 in FAULT state for bi-stable and for PS = 1, it is inverted. All the diagrams and tables are related to the default configuration selection: switching mode (FCCU_CFG.SM = 0b). In case of inverted polarity (FCCU_CFG.PS = 1b) all the values on the EOUT output pins are inverted.

Two modes can be programmed to define the EOUT protocol transitions in dual-rail or time-switching mode:

- slow switching mode: no EOUT frequency violation during the FCCU state transition (NORMAL to ERROR or viceversa and CONFIG to NORMAL). The EOUT protocol transition occurs after a max delay equal to the duration of the semi-period of the EOUT frequency.
- fast switching mode: The EOUT protocol transition (NORMAL to ERROR or viceversa and CONFIG to NORMAL) occurs immediately. A pulse with the minimum duration corresponding to 16 MHz/1024 (IRCOSC clock) period can occurs in fast switching mode. It implies a frequency violation of the EOUT protocol.

The EOUT frequency is generated by dividing IRCOSC clock by a fixed factor of 2^{18} . With an IRCOSC clock of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and the IRCOSC clock detecting the edge transition of the EOUT protocol in dual-rail or time-switching mode.

In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before. If SW configures the error pins to OK(1) and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and error pins indicate NOK ie. an incoming fault is not masked when SW has set the error pin to high. During the T_min by a non-SW fault, the FCCU FSM moves independently of this pin state (low) and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in and it is not possible to set the pins to OK by SW moving FCCU to CONFIG state, as long as this timer is running. No SW intervention is needed to bring the pin from the low state. The SW can bring the pin back to OK state by clearing the faults and waiting for the T_min to expire, after which the FCCU automatically enters NORMAL state and the error pin indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

Resets (except the power on reset and software bit FCCU_RST in RGM_PRST5 should not influence the state of the failure indication ie pins.

Note: The transition from error to config state is not possible but is shown to display the behavior in all four states — reset, normal, error, and config.

Table 1862. FCCU error pin behavior on STATE transitions⁽¹⁾

Fault signaling state	Transition from NORMAL to FAULT state			NORMAL+CONFIG states			Transition from NORMAL to ALARM state		
	with Fault disabled enabled						with Fault disabled enabled		
	Internal Error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA OK	NA 1	NA OK	OK	1	OK	NA OK	NA 1	NA OK
Enabled	NA NOK	NA 1	NA NOK	OK	1	OK	NA OK	NA 1	NA OK

1. Table valid only after SW has written fccu_set_after_reset bit to 1.

Note: FCCU remains in NORMAL state when fault is disabled hence, “NA” (Not Applicable) appears in the above table.

81.5.9.1 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual-rail encoded circuits use two wires to carry each bit. The encoding scheme is given in [Table 1863](#) and the related timing diagram is given in [Figure 2012](#) and [Figure 2013](#).

Table 1863. Dual-rail encoding

Logical state	Dual-rail encoding (output pins fccu_eout[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	00	toggling
faulty	11	
reset	Hi impedance	no toggling
configuration	same as NORMAL	toggling

As long as FCCU is in NORMAL or ALARM state, output will show “non-faulty” signal. Output pins #0, #1 will toggle between 01 and 10 with a given frequency of 61 Hz.

In the RESET phase the output pins are set as “high impedance”.

Note: [Figure 2012](#) and [Figure 2013](#) are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

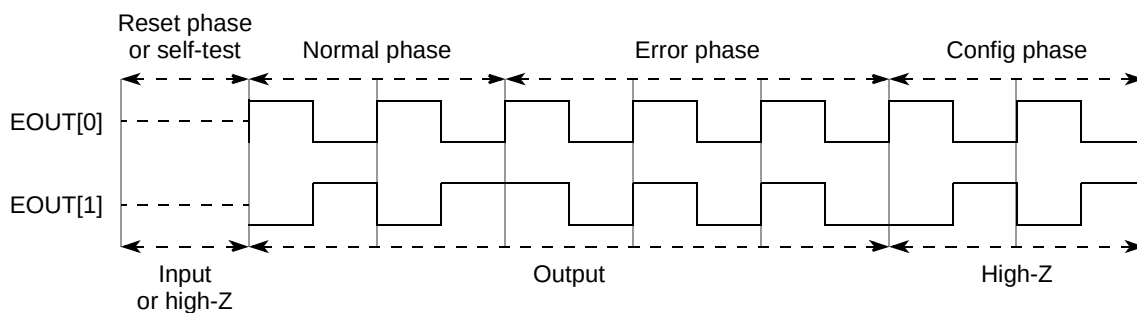


Figure 2012. Dual-rail protocol (slow switching mode)

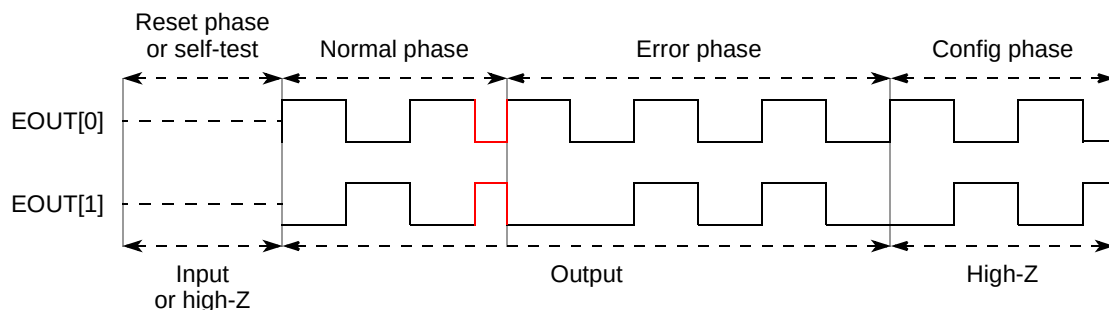


Figure 2013. Dual-rail protocol (fast switching mode)

81.5.9.2 Time-switching protocol

The encoding scheme is given in [Table 1864](#) and the related timing diagram is given in [Figure 2014](#).

Table 1864. Time-switching encoding

Logical state	Time-switching encoding (output pins fccu_eout[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	10	no toggling
reset	high-Z	no toggling
configuration	same as NORMAL	toggling

As long as FCCU is in NORMAL or ALARM state, outputs will show “non-faulty” signal. Output pins #0, #1 will toggle between 01 and 10 with frequency of 61 Hz.

In the FAULT state, the output pin fccu_eout[0] is set as low.

In Time Switching mode the second output (fccu_eout[1]) is the inverted signal of first output (fccu_eout[0]).

In the RESET phase the output pins are set as “high impedance”.

Note: *Figure 2014 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.*

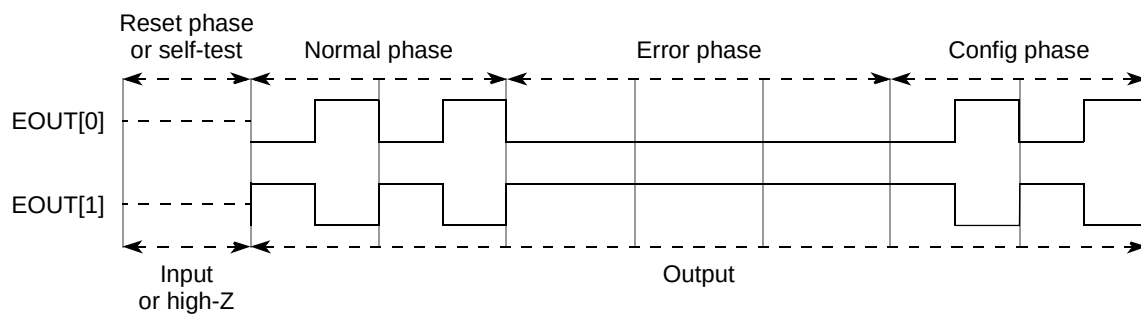


Figure 2014. Time-switching protocol

81.5.9.3 Bi-stable protocol

The encoding scheme is given in [Table 1865](#) and the related timing diagram is given in [Figure 2015](#).

Table 1865. Bi-stable encoding

Logical state	Bi-stable encoding (output pins fccu_eout[1:0])	Note
non-faulty	01	no toggling
faulty	10	no toggling
reset	high-Z	no toggling
configuration	same as NORMAL	no toggling

In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, “no-faulty” state is indicated. In Bi-stable mode the second output (fccu_eout[1]) is the inverted signal of first output (fccu_eout[0]).

In the RESET phase the output pins are set as “high impedance”.

Note: *Figure 2015 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.*

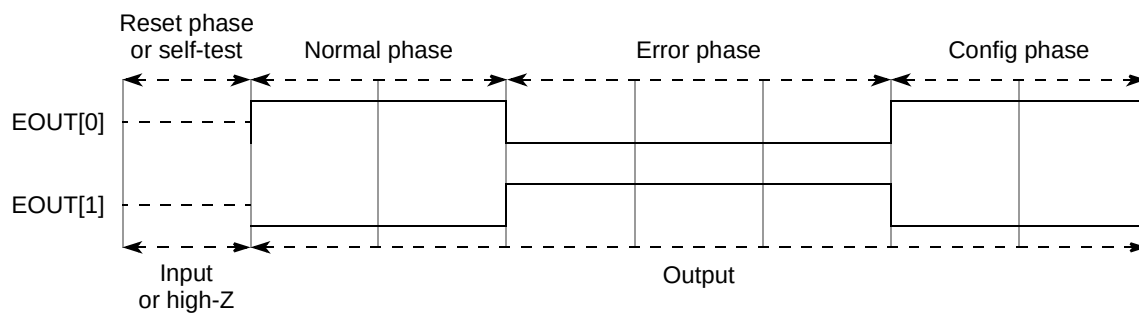
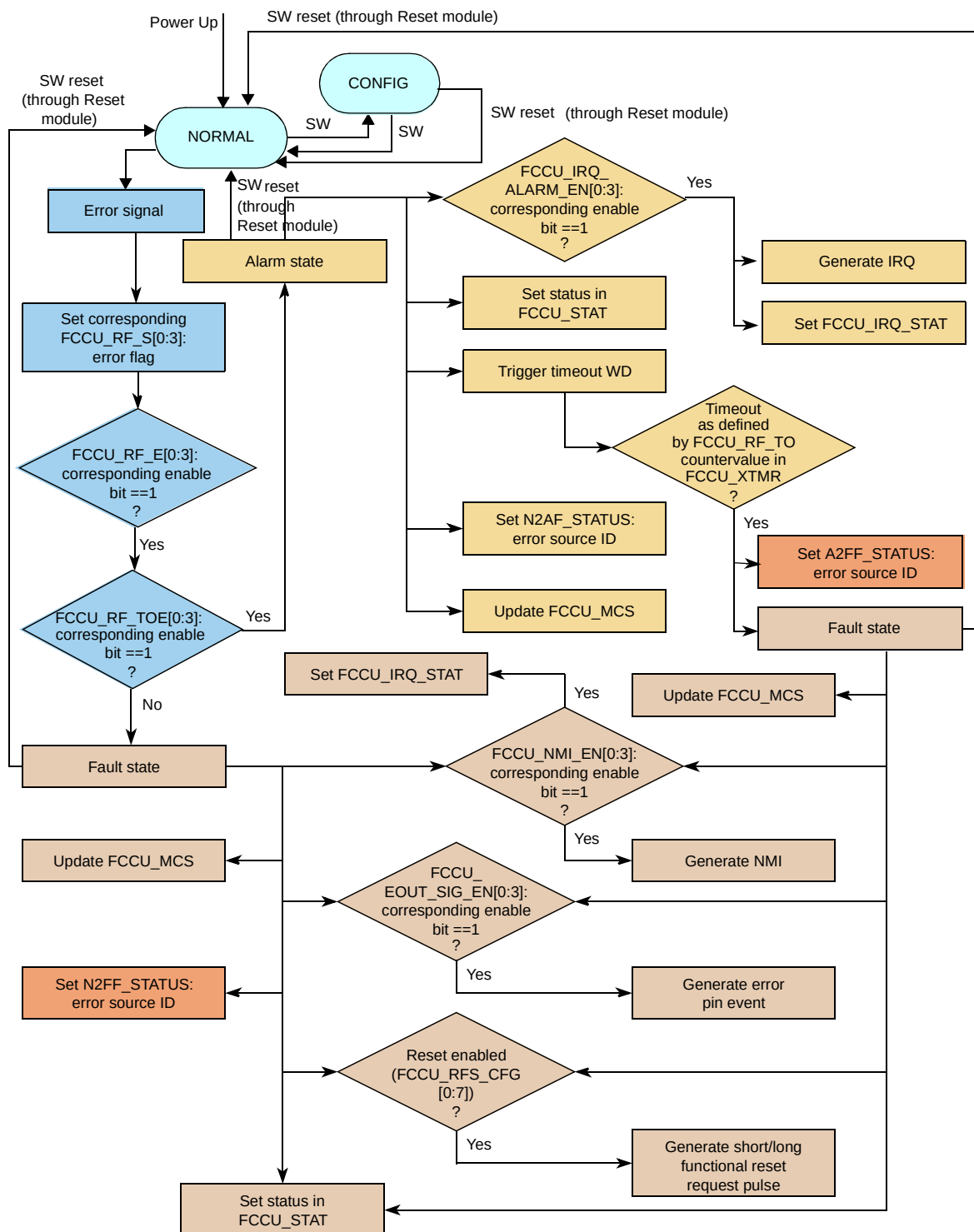


Figure 2015. Bi-stable protocol



81.6 Register description

The FCCU registers are listed in [Table 1866](#). Any address offset not explicitly mentioned in [Table 1866](#) is reserved.

All the registers accessible in write mode only in CONFIG state (last column of the [Table 1866](#)) are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible RF failure source a different reaction shall be configurable through the use of NMI, IRQ, long/short reset selection registers as well as no reaction by disabling the former registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

Table 1866. FCCU register set

Address offset	Register	Access (1)	Register classification (2)	Location
000h	FCCU Control Register (FCCU_CTRL)	R/W always	Saf-Lat	on page 3496
004h	FCCU CTRL Key Register (FCCU_CTRLK)	W always	Saf-Relv	on page 3498
008h	FCCU Configuration Register (FCCU_CFG)	R always; W in CONFIG state only	Saf-Lat	on page 3498
01Ch	FCCU RF Configuration Register 0 (FCCU_RF_CFG0)	R always; W in CONFIG state only	Saf-Lat	on page 3500
020h	FCCU RF Configuration Register 1 (FCCU_RF_CFG1)		Saf-Lat	
024h	FCCU RF Configuration Register 2 (FCCU_RF_CFG2)		Saf-Lat	
028h	FCCU RF Configuration Register 3 (FCCU_RF_CFG3)		Saf-Lat	
04Ch	FCCU RFS Configuration Register 0 (FCCU_RFS_CFG0)	R always; W in CONFIG state only	Saf-Lat	on page 3502
050h	FCCU RFS Configuration Register 1 (FCCU_RFS_CFG1)		Saf-Lat	
054h	FCCU RFS Configuration Register 2 (FCCU_RFS_CFG2)		Saf-Lat	
058h	FCCU RFS Configuration Register 3 (FCCU_RFS_CFG3)		Saf-Lat	
05Ch	FCCU RFS Configuration Register 4 (FCCU_RFS_CFG4)		Saf-Lat	
060h	FCCU RFS Configuration Register 5 (FCCU_RFS_CFG5)		Saf-Lat	
064h	FCCU RFS Configuration Register 6 (FCCU_RFS_CFG6)		Saf-Lat	
068h	FCCU RFS Configuration Register 7 (FCCU_RFS_CFG7)		Saf-Lat	

Table 1866. FCCU register set (continued)

Address offset	Register	Access (1)	Register classification (2)	Location
080h	FCCU RF Status Register 0 (FCCU_RF_S0)	R/W always	Saf-Lat	on page 3503
084h	FCCU RF Status Register 1 (FCCU_RF_S1)		Saf-Lat	
088h	FCCU RF Status Register 2 (FCCU_RF_S2)		Saf-Lat	
08Ch	FCCU RF Status Register 3 (FCCU_RF_S3)		Saf-Lat	
090h	FCCU RF Key Register (FCCU_RFK)	W always	Saf-Relv	on page 3505
094h	FCCU RF Enable Register 0 (FCCU_RF_E0)	R always; W in CONFIG state only	Saf-Lat	on page 3505
098h	FCCU RF Enable Register 1 (FCCU_RF_E1)		Saf-Lat	
09Ch	FCCU RF Enable Register 2 (FCCU_RF_E2)		Saf-Lat	
0A0h	FCCU RF Enable Register 3 (FCCU_RF_E3)		Saf-Lat	
0A4h	FCCU RF Time-out Enable Register 0 (FCCU_RF_TOE0)	R always; W in CONFIG state only	Saf-Lat	on page 3506
0A8h	FCCU RF Time-out Enable Register 1 (FCCU_RF_TOE1)		Saf-Lat	
0ACh	FCCU RF Time-out Enable Register 2 (FCCU_RF_TOE2)		Saf-Lat	
0B0h	FCCU RF Time-out Enable Register 3 (FCCU_RF_TOE3)		Saf-Lat	
0B4h	FCCU RF Time-out Register (FCCU_RF_TO)	R always; W in CONFIG state only	Saf-Lat	on page 3507
0B8h	FCCU CFG Timeout Register (FCCU_CFG_TO)	R always; W in all states except CONFIG	Saf-Lat	on page 3508
0BCh	FCCU IO Control Register (FCCU_EINOUT)	R/W always	Saf-Lat	on page 3509
0C0h	FCCU Status Register (FCCU_STAT)	R only	Saf-Lat	on page 3510
0C4h	FCCU NA Freeze Status Register (N2AF_STATUS)	R/W always	No-Saf	on page 3512
0C8h	FCCU AF Freeze Status Register (A2FF_STATUS)	R/W always	No-Saf	on page 3513
0CCh	FCCU NF Freeze Status Register (N2FF_STATUS)	R/W always	No-Saf	on page 3514
0D0h	FCCU FA Freeze Status Register (F2A_STATUS)	R/W always	No-Saf	on page 3515
0DCh	FCCU RF Fake Register (FCCU_RFF)	R/W always	No-Saf	on page 3516
0E0h	FCCU IRQ Status Register (FCCU_IRQ_STAT)	R/W always	Saf-Relv	on page 3517

Table 1866. FCCU register set (continued)

Address offset	Register	Access (1)	Register classification (2)	Location
0E4h	FCCU IRQ Enable Register (FCCU_IRQ_EN)	R/W always	Saf-Lat	on page 3518
0E8h	FCCU XTMR Register (FCCU_XTMR)	R/W always	No-Saf	on page 3519
0ECh	FCCU MCS Register (FCCU_MCS)	R only	Saf-Lat	on page 3520
0F0h	FCCU Transient Lock Register (FCCU_TRANS_LOCK)	W always	Saf-Lat	on page 3521
0F4h	FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)	W always	Saf-Lat	on page 3522
0F8h	FCCU Delta T Register (FCCU_DELTA_T)	R always; W in CONFIG state only	Saf-Lat	on page 3522
0FCh	FCCU IRQ Alarm Enable Register 0 (FCCU_IRQ_ALARM_EN0)	R always; W in CONFIG state only	Saf-Lat	on page 3523
100h	FCCU IRQ Alarm Enable Register 1 (FCCU_IRQ_ALARM_EN1)		Saf-Lat	
104h	FCCU IRQ Alarm Enable Register 2 (FCCU_IRQ_ALARM_EN2)		Saf-Lat	
108h	FCCU IRQ Alarm Enable Register 3 (FCCU_IRQ_ALARM_EN3)		Saf-Lat	
10Ch	FCCU NMI Enable Register 0 (FCCU_NMI_EN0)	R always; W in CONFIG state only	Saf-Lat	on page 3524
110h	FCCU NMI Enable Register 1 (FCCU_NMI_EN1)		Saf-Lat	
114h	FCCU NMI Enable Register 2 (FCCU_NMI_EN2)		Saf-Lat	
118h	FCCU NMI Enable Register 3 (FCCU_NMI_EN3)		Saf-Lat	
11Ch	FCCU EOUT Signaling Enable Register 0 (FCCU_EOUT_SIG_EN0)	R always; W in CONFIG state only	Saf-Lat	on page 3524
120h	FCCU EOUT Signaling Enable Register 1 (FCCU_EOUT_SIG_EN1)		Saf-Lat	
124h	FCCU EOUT Signaling Enable Register 2 (FCCU_EOUT_SIG_EN2)		Saf-Lat	
128h	FCCU EOUT Signaling Enable Register 3 (FCCU_EOUT_SIG_EN3)		Saf-Lat	

1. In this column, R/W = read/write, R = read-only, and W = write-only.

2. No-Saf = Not Safety Relevant, Saf-Relv = Safety Relevant, Saf-Lat = Safety Latent

81.6.1 FCCU Control Register (FCCU_CTRL)

The FCCU_CTRL register allows to execute the following operations:

- moving the FCCU state from the NORMAL state into the CONFIG state
- moving the FCCU state from the CONFIG state into the NORMAL state
- reading or to clear the RF status register
- reading the FCCU FSM status register
- reading or to clear the FCCU freeze registers
- reading the ALARM timer
- reading the Watchdog timer
- configure the glitch filter present on location given by FILTER_POSITION parameter

Some critical operations requires a key as defined in the FCCU_CTRLK register.

Figure 2017. FCCU Control Register (FCCU_CTRL)

Offset: 000h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FILTER_BYPASS	FILTER_WIDTH		0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	DEBUG	0	OPS		0	OPR				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1867. FCCU_CTRL field descriptions

Field	Description
FILTER_BYPASS	Filter bypass 0: glitch filter not bypassed 1: glitch filter bypassed
FILTER_WIDTH	Filter width 00: filters glitches up to 50 μ s 01: filters glitches up to 75 μ s 10,11: filters glitches up to 100 μ s for 20 MHz IRC clock
DEBUG	Debug mode entry 0: Normal operation 1: Put FCCU into debug mode When ipg_debug signal is asserted and this bit is set, FCCU moves into debug state and FCCU's FSM remains in the same state.

Table 1867. FCCU_CTRL field descriptions (continued)

Field	Description
OPS	Operation status 00: Idle 01: In progress 10: Aborted 11: Successful This bit can be read and cleared (via OP15 operation) by the software.
OPR	<i>Operation run</i> 00000: No operation [OP0] 00001: Set the FCCU into the CONFIG state [OP1] 00010: Set the FCCU into the NORMAL state [OP2] 00011: Read the FCCU state (see the FCCU_STAT register) [OP3] 00100: Read the FCCU frozen status flags (see the N2AF_STATUS register) [OP4] 00101: Read the FCCU frozen status flags (see the A2FF_STATUS register) [OP5] 00110: Read the FCCU frozen status flags (see the N2FF_STATUS register) [OP6] 00111: Read the FCCU frozen status flags (see the F2AF_STATUS register) [OP7] 01000: Read the FCCU frozen status flags (see the FCCU_SCFS register) [OP8] 01001: Reserved 01010: Read the RF status register (see the FCCU_RF_S register) [OP10] 01011: Reserved 01100: RF status clear operation in progress (see the FCCU_RF_S register) [OP12] 01101: Clear the freeze status registers (see the freeze registers) [OP13] 01110: CONFIG to NORMAL FCCU state (configuration timeout) in progress [OP14] 01111: Clear the operation status (OPS=Idle) [OP15] 10000: Reserved 10001: Read the ALARM timer (see the FCCU_XTMR register) [OP17] 10010: RESERVED 10011: Read the CFG timer (see the FCCU_XTMR register) [OP19] 10100: Read the Error Pin low counter value (see Section 81.6.21, FCCU XTMR Register (FCCU_XTMR)) [OP20] 10101–11111: Reserved The SW application must not modify the OPR field (any write operation will be ignored) until the completion of the operation. Once the operation has been completed, the operation status (OPS) is set and the OPR field is automatically cleared (OPR = 000). The opcode OP11, OP12 must not be programmed. The OPR field is automatically set to OP11 or OP12 when the FCCU_RF_S registers are cleared by a write-clear operation into the related register. The opcode OP14 must not be programmed. The OPR field is automatically set to OP14 when the timeout occurs (FCCU_CFG_TO) during the configuration procedure => the FCCU state is automatically forced in NORMAL mode setting the default configuration. In this phase any write operation to the FCCU configuration registers is inhibited. The reserved operations (OP21 -OP30) are forbidden. In case of execution, they return an ABORT response without any side effect. The ABORT response occurs in the following cases: – wrong access (missing or wrong key) to the FCCU_RF_S register (clear operation OP12) – wrong access (missing or wrong key) to the FCCU_CTRL register (OP1, OP2 operation) – OP1 (CONFIG command) execution when FCCU state != NORMAL or configuration locked – OP21-OP30 (RESERVED operations) execution

81.6.2 FCCU CTRL Key Register (FCCU_CTRLK)

The FCCU_CTRLK register implements the key access for the operations OP1, OP2 according to the following sequence:

1. Write the key into the FCCU_CTRLK register.
2. Write the FCCU_CTRL register (operations OP1 or OP2).

Note: *There should not be any other operation in between the above steps.*

The FCCU_CTRLK register is not readable, a 0000_0000h value is always returned in case of read operation. The key must be written by a word (32 bits) data write operation.

Figure 2018. FCCU CTRL Key Register (FCCU_CTRLK)

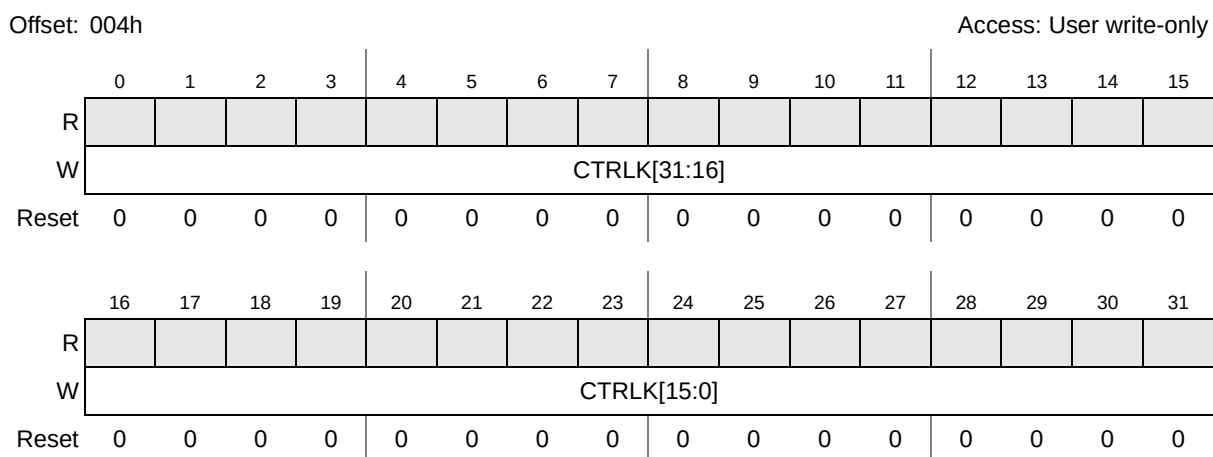


Table 1868. FCCU_CTRLK field descriptions

Field	Description
CTRLK	Control register key = 913756AFh: Key for the operation OP1 = 825A132Bh: Key for the operation OP2

81.6.3 FCCU Configuration Register (FCCU_CFG)

The FCCU_CFG register defines the global configuration for the FCCU module.

This register is writable only in the CONFIG state.

Figure 2019. FCCU Configuration Register (FCCU_CFG)

Offset: 008h

Access: User
read/write⁽¹⁾

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	FCCU_SET_AFTER_RESET	FCCU_SET_CLEAR		0	0	Reserved			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	Reserved		OD	0	SM	PS	FOM	Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only in the CONFIG state.

Table 1869. FCCU_CFG field descriptions

Field	Description
FCCU_SET_AFTER_RESET	This bit controls the enable of the o/p error pin after reset lifts 0: FCCU's Error indication stops functioning and error pins are in HI-Z state. 1: write to start FCCU functioning. After power-on the error out pins shall be in high impedance. They will go to normal state only on software request ie. this bit is set to '1'. It is SW's responsibility to write this bit to 1 else the error pin stays in High-Z state after reset lifts and FCCU is not functioning.
FCCU_SET_CLEAR	Error pin state can be controlled by these bits 01: Error Pin goes in Faulty state. 11 write: Error Pin goes OK state (write not possible when FCCU already in the T min timer phase (entered into a fault or by SW writing fccu_set_clear bit field to 01)). 00, 10: FCCU acts independent of above SW control. These bits clear(0) and set(1) the error pin. Higher priority is of the FCCU_SET_AFTER_RESET bit's capability to lead the error pins to Hi-Z. Switch to software configuration (01, 11) comes into effect when FCCU FSM leaves CONFIG state. After this is in effect, software maintains priority over FCCU FSM. The switch back to FSM control is again done by writing into these fields with 00 or 10 by going to CONFIG state.

Table 1869. FCCU_CFG field descriptions (continued)

Field	Description
OD	Open Drain Mechanism to select between Push-pull and Open drain(OD) mode for the error indicating pin(s) 0: push-pull 1: OD
SM	<i>Switching mode</i> 0: EOUT protocol (dual-rail, time-switching) slow switching mode. 1: EOUT protocol (dual-rail, time-switching) fast switching mode. SM has no effect on the bi-stable protocol.
PS	<i>Polarity selection</i> 0: fcc_eout[1] active high, fcc_eout[0] active low 1: fcc_eout[1] active low, fcc_eout[0] active high This bit can be read and written by the software. Active state here means a state indicating FAULT and applicable when pins are non-toggling in Error phase.
FOM	<i>Fault Output Mode selection</i> 000: Dual-Rail (default state) [fccu_eout[1:0]= outputs] 001: Time Switching [fccu_eout[1:0]= output to be used] 010: Bi-Stable 011: Reserved 100: Reserved 101: Test0 (controlled by FCCU_EOUT register) [fccu_eout[0]= input, fccu_eout[1]= output] 110: Test1 (controlled by FCCU_EOUT register) [fccu_eout[0]= output, fccu_eout[1]= output] 111: Test2 (controlled by FCCU_EOUT register) [fccu_eout[0]= output, fccu_eout[1]= input] <i>Note: In Testx mode, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs on the system/IRCOSC clock.</i>

81.6.4 FCCU RF Configuration Registers (FCCU_RF_CFG[0:3])

The FCCU_RF_CFGx registers contain the configuration of each recoverable fault in terms of fault recovery management. The configuration depends on the type of signaling of a fault event. HW recoverable faults should be configured only if a previous latching stage captures and hold the physical fault otherwise the fault can be lost. All the other faults should be configured as SW fault.

These registers are writable only in the CONFIG state.

Figure 2020. FCCU RF Configuration Register [0:3] (FCCU_RF_CFG[0:3])

Offset: 01Ch–28h (4 registers)

Access: User
read/write⁽¹⁾

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾
W	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾
W	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾	RFCx ⁽²⁾

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

1. Writable only in the CONFIG state
2. See [Table 1871](#) for register offset to channel number relationship.

Table 1870. FCCU_RF_CFG[0:3] field descriptions

Field	Description
RFCx	<i>Recoverable fault configuration</i> 0: HW recoverable fault 1: SW recoverable fault The <i>recoverable</i> fault configuration defines the fault recovery mode. HW recoverable faults are self recovered (status flag clearing and related) if the root cause (input fault) has been removed. SW recoverable faults are recovered (status flag clearing) by SW clearing of the related status flag.

[Table 1871](#) shows FCCU RF configuration register channels.**Table 1871. FCCU RF configuration register channels**

Address offset	Register name	Channel range (x) (bit location [0:31])
1Ch	FCCU_RF_CFG0	RFC[31:0]
20h	FCCU_RF_CFG1	RFC[63:32]
24h	FCCU_RF_CFG2	RFC[95:64]
28h	FCCU_RF_CFG3	RFC[127:96]

81.6.5 FCCU RFS Configuration Registers (FCCU_RFS_CFG[0:7])

The FCCU_RFS_CFGx registers contain the configuration of each recoverable fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

These registers are writable only in the CONFIG state.

Figure 2021. FCCU RFS Configuration Register (FCCU_RFS_CFG[0:7])

Offset: 04Ch–068h (8 registers)

Access: User
read/write⁽¹⁾

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾	
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾		RFSCx ⁽²⁾	
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. Writable only in the CONFIG state
2. See [Table 1873](#) for register offset to channel number relationship.

Table 1872. FCCU_RFS_CFG[0:7] field descriptions

Field	Description
RFSCx	<i>Recoverable fault state configuration</i> 00: No reset reaction 01: Short functional reset request pulse (FAULT state reaction) 10: Long functional reset request pulse (FAULT state reaction) 11: No reset reaction

[Table 1873](#) shows FCCU RFS configuration register channels.

Table 1873. FCCU RFS configuration register channels

Address offset	Register name	Channel range (x)
4Ch	FCCU_RFS_CFG0	RFSCx[15:0]
50h	FCCU_RFS_CFG1	RFSCx[31:16]
54h	FCCU_RFS_CFG2	RFSCx[47:32]
58h	FCCU_RFS_CFG3	RFSCx[63:48]
5Ch	FCCU_RFS_CFG4	RFSCx[79:64]
60h	FCCU_RFS_CFG5	RFSCx[95:80]

Table 1873. FCCU RFS configuration register channels

Address offset	Register name	Channel range (x)
64h	FCCU_RFS_CFG6	RFSCx[111:96]
68h	FCCU_RFS_CFG7	RFSCx[127:112]

81.6.6 FCCU RF Status Register (FCCU_RF_S[0:3])

The FCCU_RF_Sx register contains the latched fault indication collected from the recoverable fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the RF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_RF_Ex register and the respective enable bit for the time-out is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_RF_Ex register and the respective enable bit for the time-out is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the time-out (FCCU_TO register) is elapsed before to recovery the fault.

The time-out is stopped only when the FCCU returns in the NORMAL state.

The FCCU_RF_Sx register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_RF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_RFK register.
2. Clear the status (flag) bit RFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_RF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root (see [Section 81.5.7, FAULT interface](#)).

Note: *There should not be any other operation in between the above steps.*

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_RF_Sx read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_RF_Sx register.

In case of reconfiguration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_RF_Sx must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following errors are ignored:

- to write a wrong key into the FCCU_RFK register
- to attempt to clear a HW recoverable error

Figure 2022. FCCU RF Status Register (FCCU_RF_S[0:3])

Offsets: 080h–08Ch (4 registers)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾	RFSx ⁽¹⁾
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See [Table 1875](#) for register offset to channel number relationship.

Table 1874. FCCU_RF_S[0:3] field descriptions

Field	Description
RFSx	Recoverable fault status 0: No “recoverable” fault latched 1: Recoverable fault latched The status bits related to the recoverable fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed.

[Table 1875](#) shows FCCU RF status register channels.

Table 1875. FCCU RF status register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
80h	FCCU_RF_S0	RFS[31:0]
84h	FCCU_RF_S1	RFS[63:32]
88h	FCCU_RF_S2	RFS[95:64]
8Ch	FCCU_RF_S3	RFS[127:96]

81.6.7 FCCU RF Key Register (FCCU_RFK)

The FCCU_RFK register implements the key access to clear the status flags of the FCCU_RF_Sx register.

The status bits of the FCCU_RF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_RFK register.
2. Clear the status (flag) bit RFSx.

The key must be written for each FCCU_RF_Sx clear operation.

Note: There should not be any other operation in between the above steps.

The FCCU_RFK register is not readable; a 0000_0000h value is always returned in case of read operation.

Figure 2023. FCCU RF Key Register (FCCU_RFK)

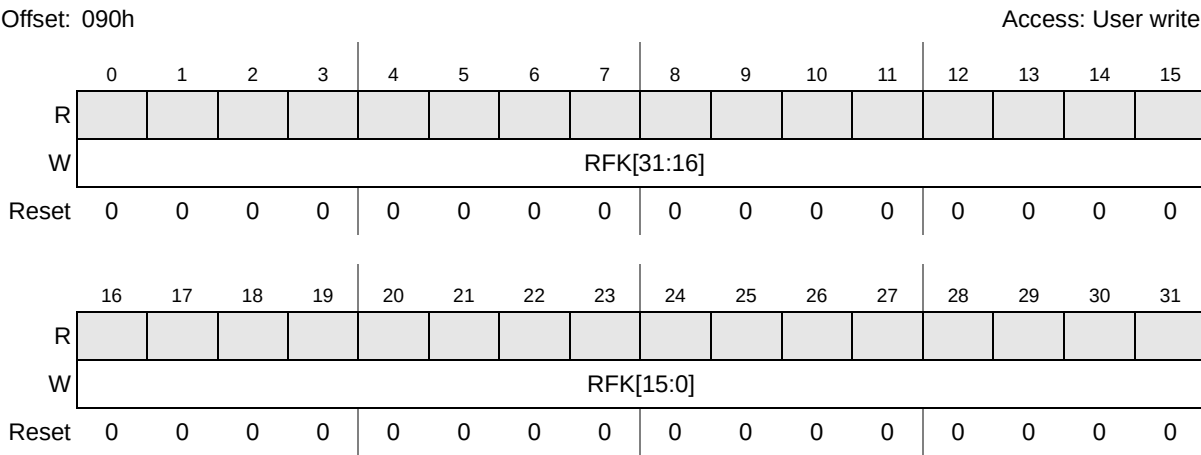


Table 1876. FCCU_RFK field descriptions

Field	Description
RFK	recoverable fault key = AB34_98FEh

81.6.8 FCCU RF Enable Registers (FCCU_RF_E[0:3])

The FCCU_RF_Ex registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_RF_Sx register is anyway set (for debugging purposes), only the reaction is masked.

These registers are writable only in the CONFIG state.

Figure 2024. FCCU RF Enable Register (FCCU_RF_E[0:3])

Offsets: 094h–0A0h (4 registers)

Access: User
read/write⁽¹⁾

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)	RFE _x (2)
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

1. Writable only in the CONFIG state
2. See [Table 1878](#) for register offset to channel number relationship.

Table 1877. FCCU_RF_E[0:3] field descriptions

Field	Description
RFE _x	Recoverable fault enable 0: No actions following the respective recoverable fault assertion 1: FCCU moves to ALARM or FAULT state

[Table 1878](#) shows FCCU RFE enable register channels.

Table 1878. FCCU RFE enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
94h	FCCU_RF_E0	RFE[31:0]
98h	FCCU_RF_E1	RFE[63:32]
9Ch	FCCU_RF_E2	RFE[95:64]
A0h	FCCU_RF_E3	RFE[127:96]

81.6.9 FCCU RF Timeout Enable Registers (FCCU_RF_TOE[0:3])

The FCCU_RF_TOEx registers enable a transition from the NORMAL state into the ALARM state if the respective recoverable fault is enabled (RFE_x and RFTOEx are set). In case the respective time-out is disabled (RFTOEx is cleared) and the recoverable fault is enabled (RFE_x is set) the FCCU moves into the FAULT state if the related recoverable fault is asserted. The timer (preset with the time-out value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the time-out the FCCU moves from the ALARM state to the FAULT state.

These registers are writable only in the CONFIG state.

Figure 2025. FCCU RF Timeout Enable Register (FCCU_RF_TOE[0:3])

Offsets: 0A4h–0B0h (4 registers)

Access: User
read/write⁽¹⁾

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾	RFTOEx ⁽²⁾
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

1. Writable only in the CONFIG state
2. See [Table 1880](#) for register offset to channel number relationship.

Table 1879. FCCU_RF_TOE[0:3] field descriptions

Field	Description
RFTOEx	Fault time-out enable 0: FCCU moves into the FAULT state if the respective fault is enabled 1: FCCU moves into the ALARM state if the respective fault is enabled

[Table 1880](#) shows FCCU RF timeout enable register channels.

Table 1880. FCCU RF timeout enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
A4h	FCCU_RF_TOE0	RFTOE[31:0]
A8h	FCCU_RF_TOE1	RFTOE[63:32]
ACh	FCCU_RF_TOE2	RFTOE[95:64]
B0h	FCCU_RF_TOE3	RFTOE[127:96]

81.6.10 FCCU RF Time-out Register (FCCU_RF_TO)

The FCCU_RF_TO register defines the preset value of the timer for the recovery of the recoverable faults (enabled). Once FCCU enters in ALARM state, following the assertion of a recoverable fault enabled (RFEx and RFTOEx are set), the timer starts the count down.

If the fault is not recovered within the time-out the FCCU moves from the ALARM state to the FAULT state.

This register is writable only in the CONFIG state.

Figure 2026. FCCU RF Time-out Register (FCCU_RF_TO)

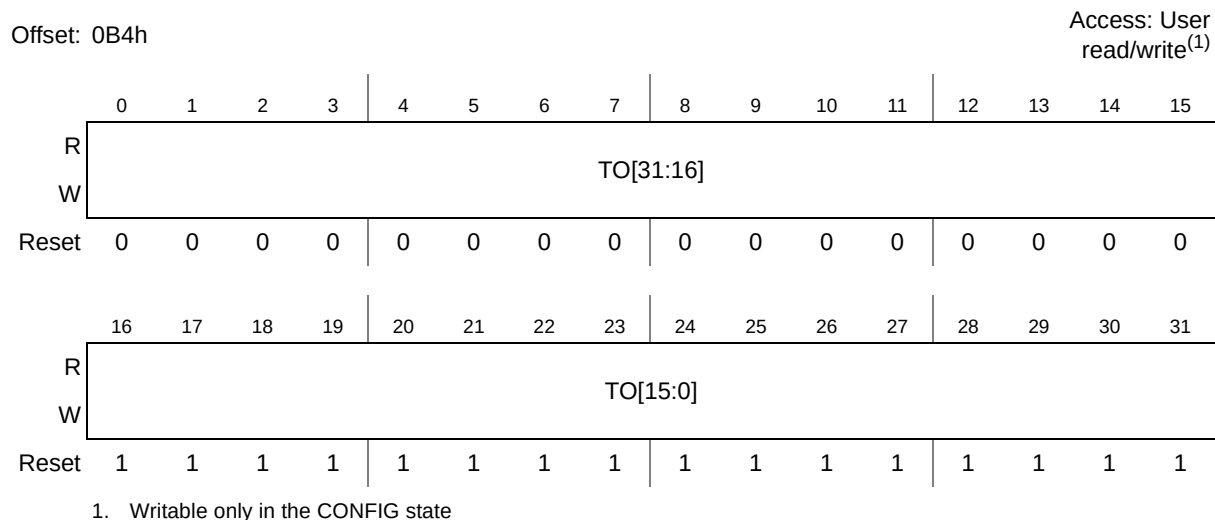


Table 1881. FCCU_RF_TO field descriptions

Field	Description
TO	<i>Recoverable fault timeout</i> $\text{Timeout} = (\text{TO}) \times T_{\text{RC16MHz}}$

81.6.11 FCCU CFG Timeout Register (FCCU_CFG_TO)

The FCCU_CFG_TO register defines the preset value of the watchdog timer for the recovery from the CONFIG state. Once FCCU enters in CONFIG state, following a SW request (OP1 opcode), the watchdog timer is initialized and starts the countdown if the reset is not asserted.

If the configuration is not completed within the time-out, the FCCU moves automatically from the CONFIG state to the NORMAL state and the default values for the configuration register is restored. The watchdog time-out is clocked with the RC oscillator clock (16 MHz). The default time-out value is 4.096 ms. Longer activation of CONFIG state can lead to resets if a failure is indicated during the time the FCCU is in CONFIG state due to FOSU.

The FCCU_CFG_TO register is writable in any state excluded the CONFIG state as follows the execution of the OP1 opcode (NORMAL to CONFIG state) and until the completion of the OP2 opcode (CONFIG to NORMAL state).

In case of watchdog time-out the FCCU_CFG_TO register is not accessible until the OP14 operation (CONFIG to NORMAL) has been completed.

Figure 2027. FCCU CFG Timeout Register (FCCU_CFG_TO)

Offset: 0B8h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TO		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Table 1882 shows FCCU_CFG_TO field descriptions.

Table 1882. FCCU_CFG_TO field descriptions

Field	Description
TO	Configuration time-out $\text{Timeout} = T_{\text{RC16MHz}} \times 2^{(\text{TO} + 10)}$ 000: Time-out = 64 μs ... 111: Time-out = 8.192 ms

81.6.12 FCCU IO Control Register (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the fccu_eout[1] output level when the FCCU is configured in “Test1” or “Test0” fault output mode (FCCU_CFG.FOM)
- to control the fccu_eout[0] output level when the FCCU is configured in “Test1” or “Test2” fault output mode (FCCU_CFG.FOM)
- to observe the fccu_ein[1:0] inputs level

Table 1883 shows Bi-stable encoding.

Table 1883. Bi-stable encoding

Mode = FCCU_CFG.FOM	FCCU_EOUT[0]	FCCU_EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

Note: Due to the resynchronization stage of the EOUT interface, there is a latency of a few IRCOSC clock cycles following a write/read operation of the FCCU_EINOUT register.

Figure 2028. FCCU IO Control Register (FCCU_EINOUT)

Offset: 0BCh

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	EIN1 ¹	EIN0 ¹	0	0	EOUT1	EOUT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. fccu.fccu_ein[x] depends on the PAD configuration, i.e., internal pull-up enabled as default or pull-down and therefore EINx bit is 1 or 0 upon reset.

Table 1884. FCCU_EINOUT field descriptions

Field	Description
EIN1	EIN1: Error input 1 0: when fccu_ein[1] = 0 1: when fccu_ein[1] = 1 The EIN1 captures the respective fccu_ein[1] input signal.
EIN0	Error input 0 0: when fccu_ein[0] = 0 1: when fccu_ein[0] = 1 The EIN0 captures the fccu_ein[0] input signal.
EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => fccu_eout[1] configured in output mode) 0: force fccu_eout[1] = 0 1: force fccu_eout[1] = 1 The EOUT1 set/clear the respective fccu_eout[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a "don't-care" value.
EOUT0	Error out 0 (significant only if the FCCU_CFG.FOM = Test1 or Test2 => fccu_eout[0] configured in output mode) 0: force fccu_eout[0] = 0 1: force fccu_eout[0] = 1 The EOUT0 set/clear the respective fccu_eout[0] output signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a "don't care" value.

81.6.13 FCCU Status Register (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine (FSM) operates by the RC oscillator clock (ipg_clk_safe) asynchronous with the system clock (ipg_clk). The FCCU status read operation requires a

safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Figure 2029. FCCU Status Register (FCCU_STAT)

Offset: 0C0h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PhysicError Pin		ESTAT	STATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Table 1885. FCCU_STAT field descriptions

Field	Description
ESTAT	Current Error Pin Status. SW can write this bit with a different value. The new value is valid for 1 clock cycle, thereafter it returns to indicate the actual value being driven from FCCU. For example in non-faulty states, this bit reads 0 and can be set for one cycle to 1. Similarly in faulty state, this bit reads 1 and sw can write it to 0 and after one cycle the bit reads 1 again. 0: System is OPERATIONAL (OK). 1: System is in FAULT state. This state is taken from the FCCU Eout logic (ie., as late as possible).
PhysicErrorPin	PhysicErrorPin : Physical Error Pin's state. PhysicErrorPin[0] corresponds to error pin 0 and PhysicErrorPin[1] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0: Error Pin is at logical 0 1: Error Pin is at logical 1 During Fault state, a static or toggling value is observed based on selected protocol.
STATUS	FCCU Status 000: NORMAL state 001: CONFIG state 010: ALARM state 011: FAULT state Other: UNKNOWN state

81.6.14 FCCU NA Freeze Status Register (N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the “recoverable” source (fccu_rf[x]) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Figure 2030. FCCU NA Freeze Status Register (N2AF_STATUS)

Offset: 0C4h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	NAFS ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can read the bits in this field, and you can write “0” to them. Writing “1” has no effect.

Table 1886. N2AF_STATUS field descriptions

Field	Description
NAFS	<i>Normal to Alarm Frozen Status</i> 00h: No transition from NORMAL to ALARM state 01h: NORMAL to ALARM state transition cause => fccu_rf[0] fault 02h: NORMAL to ALARM state transition cause => fccu_rf[1] fault 03h: NORMAL to ALARM state transition cause => fccu_rf[2] fault ... 80h: NORMAL to ALARM state transition cause => fccu_rf[127] fault FFh: NORMAL to ALARM state transition cause => multiple fccu_rf[] faults

81.6.15 FCCU AF Freeze Status Register (A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the time-out trigger (recoverable fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Figure 2031. FCCU AF Freeze Status Register (A2FF_STATUS)

Offset: 0C8h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	AF_SRC ⁽¹⁾		AFFS ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can read the bits in this field, and you can write "0" to them. Writing "1" has no effect.

Table 1887. A2FF_STATUS field descriptions

Field	Description
AF_SRC	<i>AF_SRC: Fault source</i> 00: No fault 01: Reserved 10: Recoverable fault 11: Multiple and/or recoverable faults
AFFS	<i>Alarm to Fault Frozen Status</i> 00h: No transition from ALARM to FAULT state 01h: ALARM to FAULT state transition cause => fccu_rf[0] fault time-out fault 02h: ALARM to FAULT state transition cause => fccu_rf[1] fault time-outfault 03h: ALARM to FAULT state transition cause => fccu_rf[2] fault time-outfault ... 80h: ALARM to FAULT state transition cause => fccu_rf[127] fault time-outfault FFh: ALARM to FAULT state transition cause => multiple fccu_rf[] faults time-out faults

81.6.16 FCCU NF Freeze Status Register (N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the recoverable fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Figure 2032. FCCU NF Freeze Status Register (N2FF_STATUS)

Offset: 0CCh

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	NF_SRC ⁽¹⁾		NFFS ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can read the bits in this field, and you can write "0" to them. Writing "1" has no effect.

Table 1888. N2FF_STATUS field descriptions

Field	Description
NF_SRC	<i>NF_SRC: Fault source</i> 00: No fault 01: Reserved 10: Recoverable fault 11: Multiple and recoverable faults
NFFS	<i>Normal to Fault Frozen Status</i> 00h: No transition from NORMAL to FAULT state 01h: NORMAL to FAULT state transition cause => fccu_rf[0] fault 02h: NORMAL to FAULT state transition cause => fccu_rf[1] fault 03h: NORMAL to FAULT state transition cause => fccu_rf[2] fault ... 80h: NORMAL to FAULT state transition cause => fccu_rf[127] fault FFh: NORMAL to FAULT state transition cause => multiple fccu_rf[] faults

81.6.17 FCCU FA Freeze Status Register (F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the "recoverable fault" source (fccu_rf[x]) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the F2AF_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1.

Set the OP13 operation into the FCCU_CTRL.OPR field.
2.

Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Figure 2033. FCCU FA Freeze Status Register (F2A_STATUS)

Offset: 0D0hAccess: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FAFS ⁽¹⁾							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can read the bits in this field, and you can write “0” to them. Writing “1” has no effect.

Table 1889. F2AF_STATUS field descriptions

Field	Description
FAFS	<i>Fault to Normal Frozen Status</i> 00h: No transition from FAULT to NORMAL state 01h: FAULT to ALARM state transition cause => fccu_rf[0]fault 02h: FAULT to ALARM state transition cause => fccu_rf[1]fault 03h: FAULT to ALARM state transition cause => fccu_rf[2]fault ... 80h: FAULT to ALARM state transition cause => fccu_rf[127]fault FFh: FAULT to ALARM state transition cause => multiple fccu_rf[]faults

81.6.18 FCCU RF Fake Register (FCCU_RFF)

The FCCU_RFF register contains a unique code to set a “recoverable fault” in mutually exclusive mode by the external FAULT interface signal setting. It allows the SW emulation of the recoverable faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The fault injection mechanism is optional and requires a proper management as described in [Section 81.5.7, FAULT interface](#). The reaction following a fake recoverable fault can be masked. The FCCU_RFF is a write-only register with a set of codes corresponding to each recoverable fault injection.



Figure 2034. FCCU RF Fake Register (FCCU_RFF)

Offset: 0DCh

Access: User write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0							
W										FRFC						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1890. FCCU_RFF field descriptions

Field	Description
FRFC	FRFC: <i>Fake recoverable fault code</i> 00h: Fake recoverable fault injection at recoverable fault source 0 01h: Fake recoverable fault injection at recoverable fault source 1 02h: Fake recoverable fault injection at recoverable fault source 2 ... 7Fh: Fake recoverable fault injection at recoverable fault source 127 <i>Note: Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results.</i>

81.6.19 FCCU IRQ Status Register (FCCU_IRQ_STAT)

The FCCU_IRQ_STAT register defines the FCCU interrupt status register related to the following events:

- Configuration time-out error
- Alarm interrupt
- NMI interrupt

The external interrupt line “irq_misc” is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Figure 2035. FCCU IRQ Status Register (FCCU_IRQ_STAT)

Offset: 0E0h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	Reserved	Reserved	NMI_STAT	ALRM_STAT	CFG_TO_STAT
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1891. FCCU_IRQ_STAT field descriptions

Field	Description
NMI_STAT	NMI Interrupt Status 0: NMI interrupt is OFF 1: NMI interrupt is ON
ALRM_STAT	Alarm Interrupt Status 0: Alarm interrupt is OFF 1: Alarm interrupt is ON
CFG_TO_STAT	Configuration Time-out Status 0: No configuration time-out error 1: Configuration time-out error

81.6.20 FCCU IRQ Enable Register (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following events:

Configuration time-out error

The external interrupt line “irq_misc” is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

This register is writable only in the CONFIG state.

Figure 2036. FCCU IRQ Enable Register (FCCU_IRQ_EN)

Offset: 0E4h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CFG_TO_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1892. FCCU_IRQ_EN field descriptions

Field	Description
CFG_TO_IEN	Configuration Time-out Interrupt Enable 0: Configuration time-out interrupt disabled 1: Configuration time-out interrupt enabled

81.6.21 FCCU XTMR Register (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer or EOUT pin low counter. These timers are clocked on the IRCOSC clock.

ETMR is the EOUT pin in logic 0 counter. After EOUT has been set to low (by SW or by a real fault), it goes high at the end of the T_min period from the last fault trigger.

The SW application executes the timer read operation by the following sequence:

- Set any of the following operations into the FCCU_CTRL.OPR field:
 - OP17
 - OP19
- OP20 Wait for the completion of the operation (FCCU_CTRL.OPS field)
- Read the FCCU_XTMR register

[Table 1893](#) shows Timer state/value.

Table 1893. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh
ETMR	00000000h	00000000h	00000000h	Running

Figure 2037. FCCU XTMR Register (FCCU_XTMR)

Offset: 0E8h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	XTMR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1894. FCCU_XTMR field descriptions

Field	Description
XTMR	Alarm/Watchdog/Safe request timer The current timer value is measured in IRCOSC clock cycles. These bits can be read by the software.

81.6.22 FCCU MCS Register (FCCU_MCS)

The FCCU_MCS register contains a queue of the last four states of the MC. MCS0 is the latest one, while MCS3 is the oldest one. In addition a qualifier indicates if the FCCU is in the FAULT state when the MC state has been captured. The MC state is synchronous to the system clock and provided by a different module while the FCCU state is synchronous to the IRCOSC, therefore some uncertainty must be considered regarding the FAULT state indication.

Figure 2038. FCCU MCS Register (FCCU_MCS)

Offset: 0ECh

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0	0	MCS3				VL2	FS2	0	0	MCS2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0	0	MCS1				VL0	FS0	0	0	MCS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1895. FCCU_MCS field descriptions

Field	Description
VLx	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0: MCSx, FSx fields are not significative 1: MCSx, FSx fields are significative
FSx	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0: MCSx field captured in any state different from the FAULT state 1: MCSx field captured in FAULT state
MCSx	MCSx: <i>State of MC modules</i> The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual.

81.6.23 FCCU Transient Register (FCCU_TRANS_LOCK)

The FCCU_TRANS register is used for unlocking configuration by writing BCh. This register is available only in supervisor mode. This register is write only.

Figure 2039. FCCU Transient Register (FCCU_TRANS_LOCK)

Offset: 0F0h

Access: Supervisor write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0								
W									TRANSKEY							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1896. FCCU_TRANS_LOCK field descriptions

Field	Description
TRANSKEY	Transistion Unlocking Key value BCh: Configuration Unlocked Any other value: No Change

81.6.24 FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing FFh. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.

Figure 2040. FCCU Transient Register (FCCU_PERMNT_LOCK)

Offset: 0F4h

Access: Supervisor write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0								
W									PERMNTKEY							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1897. FCCU_PERMNT_LOCK field descriptions

Field	Description
PERMNTKEY	Transistion Locking Key value FFh: Configuration Locked Any other value: No Change

81.6.25 FCCU Delta T Register (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds. This register can be written only when the FCCU is in CONFIG state.

Figure 2041. FCCU Delta T Register (FCCU_DELTA_T)

Offset: 0F8h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	Reserved													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	DELTA_T													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1898. FCCU_DELTA_T field descriptions

Field	Description
DELTA_T	DELTA_T: Value of Delta_T in microseconds Max. value can be 10,000 microseconds (10 ms)

81.6.26 FCCU IRQ Alarm Enable Registers (FCCU_IRQ_ALARM_EN[0:3])

These registers enable the corresponding IRQ alarm.

Figure 2042. FCCU IRQ Alarm Enable Registers (FCCU_IRQ_ALARM_EN[0:3])

Offsets: 0FCh–108h (4 registers)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾
W	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾
W	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾	IRQENx ⁽¹⁾
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See [Table 1900](#) for register offset to channel number relationship.

Table 1899. FCCU_IRQ_ALARM_EN[0:3] field descriptions

Field	Description
IRQENx	IRQ alarm enable 0: Alarm is disabled for error source x. 1: Alarm is enabled for error source x.

[Table 1900](#) shows FCCU IRQ alarm enable register channels.

Table 1900. FCCU IRQ alarm enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0FCh	FCCU_IRQ_ALARM_EN0	IRQENE[31:0]
100h	FCCU_IRQ_ALARM_EN1	IRQENE[63:32]
104h	FCCU_IRQ_ALARM_EN2	IRQENE[95:64]
108h	FCCU_IRQ_ALARM_EN3	IRQENE[127:96]

81.6.27 FCCU NMI Enable Registers (FCCU_NMI_EN[0:3])

These registers enable the NMI.

Figure 2043. FCCU NMI Enable Registers (FCCU_NMI_EN[0:3])

Offsets: 10Ch–118h (4 registers)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾
W	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾
W	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾	NMIENx ⁽¹⁾
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See [Table 1902](#) for register offset to channel number relationship.

Table 1901. FCCU_NMI_EN[0:3] field descriptions

Field	Description
NMIENx	NMI enable 0: NMI is disabled for error (RF) source x. 1: NMI is enabled for error (RF) source x.

[Table 1902](#) shows FCCU NMI enable register channels.

Table 1902. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
10Ch	FCCU_NMI_EN0	NMIENE[31:0]
110h	FCCU_NMI_EN1	NMIENE[63:32]
114h	FCCU_NMI_EN2	NMIENE[95:64]
118h	FCCU_NMI_EN3	NMIENE[127:96]

81.6.28 FCCU EOUT Signaling Enable Registers (FCCU_EOUT_SIG_EN[0:3])

These registers enable the error out signaling.

Figure 2044. FCCU EOUT Signaling Enable Registers (FCCU_EOUT_SIG_EN[0:3])

Offsets: 11Ch–128h (4 registers)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾
W	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾
Reset ⁽²⁾	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾
W	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾	EOUTENx ⁽¹⁾
Reset ⁽²⁾	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See [Table 1904](#) for register offset to channel number relationship.

2. For non existing fault sources, reset is 0.

Table 1903. FCCU_EOUT_SIG_EN[0:3] field descriptions

Field	Description
EOUTENx	EOUT signaling enable 0: EOUT signaling is disabled for error (RF) source x. Error pins are as if in non-faulty state. 1: EOUT signaling is enabled for error (RF) source x.

[Table 1904](#) shows FCCU EOUT signaling enable register channels.**Table 1904. FCCU EOUT signaling enable register channels**

Address offset	Register name	Channel range (x) (bit location [0:31])
11Ch	FCCU_EOUT_SIG_EN0	EOUTENE[31:0]
120h	FCCU_EOUT_SIG_EN1	EOUTENE[63:32]
124h	FCCU_EOUT_SIG_EN2	EOUTENE[95:64]
128h	FCCU_EOUT_SIG_EN3	EOUTENE[127:96]



81.7 Register map

Table 1905 shows FCCU register map.

Table 1905. FCCU register map

Addr Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
000h	FCCU_CTRL	FIL TE R_ BY PA SS	FILTER_W IDTH		-																		DE BU G	NV ML	OPS		-	OPR						
004h	FCCU_CTRLK	CTRLK																																
008h	FCCU_CFG	Reserved									SR M	NMI M	--	--	Reserved				Reserved	Re- served	-	0	SM	PS	FOM			Reserved						
01Ch ... 028h	FCCU_RF_CFG0 ... 3	RF C31	RF C30	RF C29	RF C28	RF C27	RF C26	RF C25	RF C24	RF C23	RF C22	RF C21	RF C20	RF C19	RF C18	RF C17	RF C16	RF C15	RF C14	RF C13	RF C12	RF C11	RF C10	RF C9	RF C8	RF C7	RF C6	RF C5	RF C4	RF C3	RF C2	RF C1	RF C0	
04Ch ... 068h	FCCU_RFS_CFG0 ... 7	RFSC15		RFSC14		RFSC13		RFSC12		RFSC11		RFSC10		RFSC9		RFSC8		RFSC7		RFSC6		RFSC5		RFSC4		RFSC3		RFSC2		RFSC1		RFSC0		
080h ... 08Ch	FCCU_RF_S0...3	RF S31	RF S30	RF S29	RF S28	RF S27	RF S26	RF S25	RF S24	RF S23	RF S22	RF S21	RF S20	RF S19	RF S18	RF S17	RF S16	RF S15	RF S14	RF S13	RF S12	RF S11	RF S10	RF S9	RF S8	RF S7	RF S6	RF S5	RF S4	RF S3	RF S2	RF S1	RF S0	
090h	FCCU_RFK	RFK																																
094h ... 0A0h	FCCU_RF_E0...3	RF E31	RF E30	RF E29	RF E28	RF E27	RF E26	RF E25	RF E24	RF E23	RF E22	RF E21	RF E20	RF E19	RF E18	RF E17	RF E16	RF E15	RF E14	RF E13	RF E12	RF E11	RF E10	RF E9	RF E8	RF E7	RF E6	RF E5	RF E4	RF E3	RF E2	RF E1	RF E0	
0A4h ... 0B0h	FCCU_RF_TOE0...3	RF TO E31	RF TO E30	RF TO E29	RF TO E28	RF TO E27	RF TO E26	RF TO E25	RF TO E24	RF TO E23	RF TO E22	RF TO E21	RF TO E20	RF TO E19	RF TO E18	RF TO E17	RF TO E16	RF TO OE 15	RF TO E14	RF TO E13	RF TO E12	RF TO E11	RF TO E31 0	RF TO E9	RF TO E8	RF TO E7	RF TO E6	RF TO E5	RF TO E4	RF TO E3	RF TO E2	RF TO E1	RF TO E0	
0B4h	FCCU_RF_TO	TO																																
0B8h	FCCU_CFG_TO	-																											TO					
0BCh	FCCU_EINOUT	-																									E I N1	E I N0	-		EO UT1	EO UT0		



Table 1905. FCCU register map (continued)

Addr Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0C0h	FCCU_STAT	-																										PS TA T	ES AT A	STATUS			
0C4h	N2AF_STATUS	-																								NAFS							
0C8h	A2FF_STATUS	-																					AF_SRC			AFFS							
0CCh	N2FF_STATUS	-																					NF_SRC			NFFS							
0D0h	F2A_STATUS	-																								FAFS							
0DCh	FCCU_RFF																									FRFC							
0E0h	FCCU_IRQ_STAT																											Reserved	Reserved	NMI	ALARM	CFGT OTSTAT	
0E4h	FCCU_IRQ_EN	-																															CFGT O I EN
0E8h	FCCU_XTMR	XTMR																															
0ECh	FCCU_MCS	VL3	FS3	MCS3			VL2	FS2	MCS2			VL1	FS1	MCS1			VL0	FS0				MCS0											
0F0h	FCCU_TRANS_LOCK																									TRANSKEY							
0F4h	FCCU_PERMNT_LOCK																									PERMNTKEY							
0F8h	FCCU_DELTA_T				Reserved																		DELTA_T										
0FCh ... 108h	FCCU_IRQ_ALARM_EN0 ... 3	I RQ EN 31	I RQ EN 30	I RQ EN 29	I RQ EN 28	I RQ EN 27	I RQ EN 26	I RQ EN 25	I RQ EN 24	I RQ EN 23	I RQ EN 22	I RQ EN 21	I RQ EN 20	I RQ EN 19	I RQ EN 18	I RQ EN 17	I RQ EN 16	I RQ EN1 5	I RQ EN 14	I RQ EN 13	I RQ EN 12	I RQ EN 11	I RQ EN 10	I RQ EN 9	I RQ EN 8	I RQ EN 7	I RQ EN 6	I RQ EN 5	I RQ EN 4	I RQ EN 3	I RQ EN 2	I RQ EN 1	I RQ EN 0



Table 1905. FCCU register map (continued)

Addr Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
10Ch ... 118h	FCCU_NMI_EN0 ... 3	NMI EN 31	NMI EN 30	NMI EN 29	NMI EN 28	NMI EN 27	NMI EN 26	NMI EN 25	NMI EN 24	NMI EN 23	NMI EN 22	NMI EN 21	NMI EN 20	NMI EN 19	NMI EN 18	NMI EN 17	NMI EN 16	NMI EN1 5	NMI EN 14	NMI EN 13	NMI EN 12	NMI EN 11	NMI EN 10	NMI EN 9	NMI EN 8	NMI EN 7	NMI EN 6	NMI EN 5	NMI EN 4	NMI EN 3	NMI EN 2	NMI EN 1	NMI EN 0
11Ch ... 128h	FCCU_EOUT_SIG_EN0 ... 3	EO UT EN 31	EO UT EN 30	EO UT EN 29	EO UT EN 28	EO UT EN 27	EO UT EN 26	EO UT EN 25	EO UT EN 24	EO UT EN 23	EO UT EN 22	EO UT EN 21	EO UT EN 20	EO UT EN 19	EO UT EN 18	EO UT EN 17	EO UT EN 16	EO UT EN1 5	EO UT EN 14	EO UT EN 13	EO UT EN 12	EO UT EN 11	EO UT EN 10	EO UT EN 9	EO UT EN 8	EO UT EN 7	EO UT EN 6	EO UT EN 5	EO UT EN 4	EO UT EN 3	EO UT EN 2	EO UT EN 1	EO UT EN 0

81.8 FCCU Output Supervision Unit (FOSU)

The FOSU provides a supervision of the primary error notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after an error arrives.

The intention of the FOSU is to provide a secondary error reaction path in most cases the FCCU fails but not to needlessly propagate an error which is already handled by the FCCU into a full chip reset. Only a failed primary error reaction (i.e., FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction. It is a "static" input in the sense that it is not changing after configuration. The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted. The active edge of an incoming fault that has not been masked is captured by the FOSU. There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU.

The FOSU contains a timer with a duration of FOSU_COUNT, driven by the IRCOSC. The timer is initialized and started on a captured fault for which the FCCU is programmed to show a reaction. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FCCU does not check whether the reaction is the configured one):

- Reset
- IRQ
- Error out triggered (by FCCU or by SW)

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still 'stuck' in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted. The FOSU's failure indicator output is connected to one of the MC_RGM's 'destructive' reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as is used by the FCCU (ipg_hard_asynch_rst_clk_safe_b), which is asserted on power-on, 'destructive', and long external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

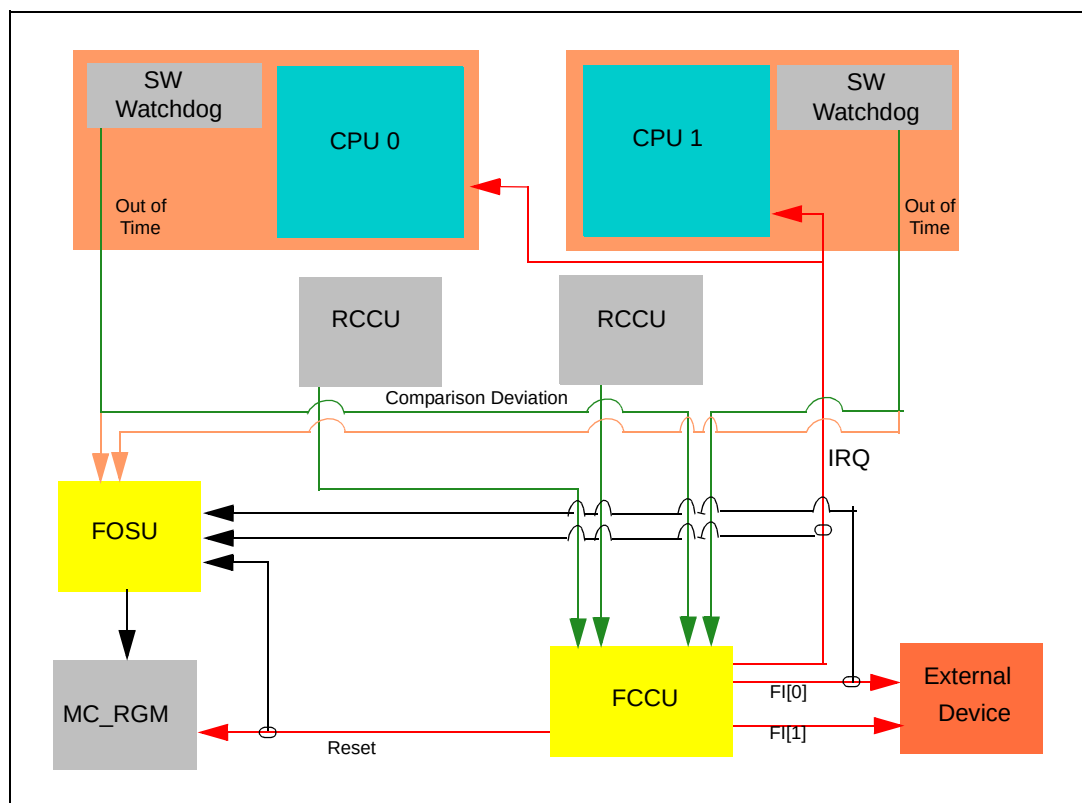


Figure 2045. FOSU connections to the FCCU and MC_RGM

It is important to remember that there will be no FCCU reaction to an error while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then reset the MCU.

81.9 Use cases and limitations

81.9.1 Misconfigurations

The following configurations are appropriate:

1. If at least one reaction to fault is enabled by default then the corresponding fault should also be enabled by default.
2. Enable a fault but disabling all reactions is not a meaningful configuration from safety point of view.
3. Disable the fault and also its reactions. Applicable when a specific fault line is not of interest in a specific application scenario.
Example: FCCU goes into ALARM state due to a failure, but the IRQ is disabled for that failure.
4. Software reset is optional.
5. If the user programs the FCCU for a higher timeout value than what is hardcoded for FOSU, then FCCU reacts later than what FOSU would be expecting and as a result, FOSU will generate destructive reset request on its timeout.

81.9.2 Recommendations to configure FCCU

1. After Power on reset, where both system and FCCU get reset, following steps could be followed to configure FCCU:
 - a) Check and clear any pending fault status
 - b) Verify FCCU is in NORMAL state, else repeat step(a) above
 - c) Configure FCCU
2. After functional reset of the system, arising out of reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
 - a) If active, wait for the Error out T_{min} to expire
 - b) Check and clear fault status
 - c) Error pin moves to “non faulty” state, once fault status is cleared and T_{min} expires
 - d) Reset FCCU by register bit in RGM. All FCCU registers and Error pin initialize to default state
 - e) Read and verify default value in FCCU_NCF_Ex to ascertain reset of FCCU
 - f) Check and clear any new pending fault status until FCCU is in NORMAL state
 - g) Reconfigure FCCU

Note: Software reset of FCCU by RGM_PRST5.FCCU_RST bit is optional since FCCU registers are not initialized during functional reset. RGM_PRST5.FCCU_RST has the same effect on FCCU as power on reset. If the bit is activated, it can change the state of the error pin, if the current state of the pin is different from default.

82 Self-Test Control Unit (STCU2)

82.1 Introduction

Self-Test Control Unit (STCU2) is a comprehensive programmable hardware module that controls the self-test sequence applied both during the Off-Line and/or On-Line conditions. It is able to manage by hardware the device's Logic Built-In Self Test (LBIST) and the SRAM/ROM Built-In Self Test (MBIST) blocks. The STCU2 includes the SSCM DCF bus to load the Self-test parameters (L/MBIST scheduling activity, LBIST setup, Unrecoverable/Recoverable faults management, CRC and MISR expected values, PLL management, and so on) from flash memory during the Off-Line Self-Test phase. This interface is able only to write the configuration parameters and start the Self-Test execution once after the STCU2 global reset has been applied. The register access by software is granted by an IPS interface with the purpose to check the results of the Off-Line Self-Test but also to load/check the execution of the On-Line Self-Test.

The STCU2 includes:

- A CRC block that can be optionally enabled to monitor the internal critical signals during the Self-Test run to increase the STCU2's Self-Checking capability
- A programmable Watchdog timer (WDG) to check the Self-Test operations (both LBIST than MBIST) have been completed within the assigned time slot or the STCU2 RUN or BYPASS bits have been programmed before Watchdog time-out
- The PLL direct control during the Off-Line Self-Test sequence
- Two interrupt lines enabled only during the On-Line Self-Test to simplify the software handling of Self-Test execution end
- Two sets of result registers to collect the L/MBIST result during Off/On-Line Self-Test separately

To increase the security capability of this module, a different Security Key sequence is also required during Off/On-Line Self-Test in order to unlock the write access to the STCU2 registers. The STCU2 also implements two power-saving mechanisms to switch off:

- The global STCU2 clock after Self-Test sequence is completed
- The register ITF clock when the WDG time-out is detected after a Double Security Key sequence is applied during the application run (available only when IPS write access is enabled otherwise the clock is off).

82.2 Main features

- Double registers interface (SSCM DCF with priority over IPS)
- SSCM DCF write one time register interface
- IPS read and write (depends on SSCM enable parameters and STCU_CFG.WRP bit) register interface
- Programmable scheduler for LBIST/MBIST execution
- LBIST concurrent/sequential execution
- MBIST concurrent/sequential execution
- Programmable LBIST delayed concurrent start
- 32-bit CRC for STCU2 Self-Checking capability (enabled by STCU_CFG.CRCEN bit)
- On-Line hardware and software Self-Test sequence abort capability
- Programmable Internal clock prescaler to reduce Internal and M/LBIST TCK clocks
- Programmable PLL direct control during Off-Line Self-test sequence
- PLL Lock signal monitoring during Off-On/Line Self Test
- L/MBIST dedicated On-Line interrupt source generation
- Programmable WDG Timer for Self-Check M/LBIST Time Execution
- Hard Coded WDG time-out to flag off-line initialization errors
- Hard Coded WDG time-out to Auto-lock STCU2 write access after Double-Key sequence
- LBIST and SRAM/ROM MBIST Off-Line status flags
- STCU2 internal errors Off-Line status flag
- LBIST and SRAM/ROM MBIST On-Line status flags
- STCU2 internal errors On-Line Status flag
- On-Line hardware and software abort status flag
- LBIST and SRAM/ROM MBIST Off-On/Line programmable failure severity (UF/RF)
- STCU2 internal errors Off-On/Line programmable failure severity (UF/RF)
- STCU2 errors (UF/RF) lines for FCCU signaling
- Redundant UF/RF generation logic to improve safety
- FCCU UF/RF faults injection mechanism
- CRCE register access to perform redundant check by software
- LBIST MISRE registers access to perform redundant check by software
- Self-Test Bypass capability
- Global write register's protection mechanism based on Double Security Key
- STCU2 Global Auto Power-saving when Self-Test is completed, bypass mode is selected or WDT time-out is detected
- STCU2 ITF/WDG clock wake-up mechanism when software application writes Double Security Key sequence (available when STCU_CFG.WRP bit is active)
- STCU2 ITF/WDG Auto Power-saving when Hard Coded WDG time-out is detected after software application writes Double Security Key sequence (available when STCU_CFG.WRP bit is active)

82.3 Block diagram

The top level diagram of the STCU2 module is given in [Figure 2046](#).

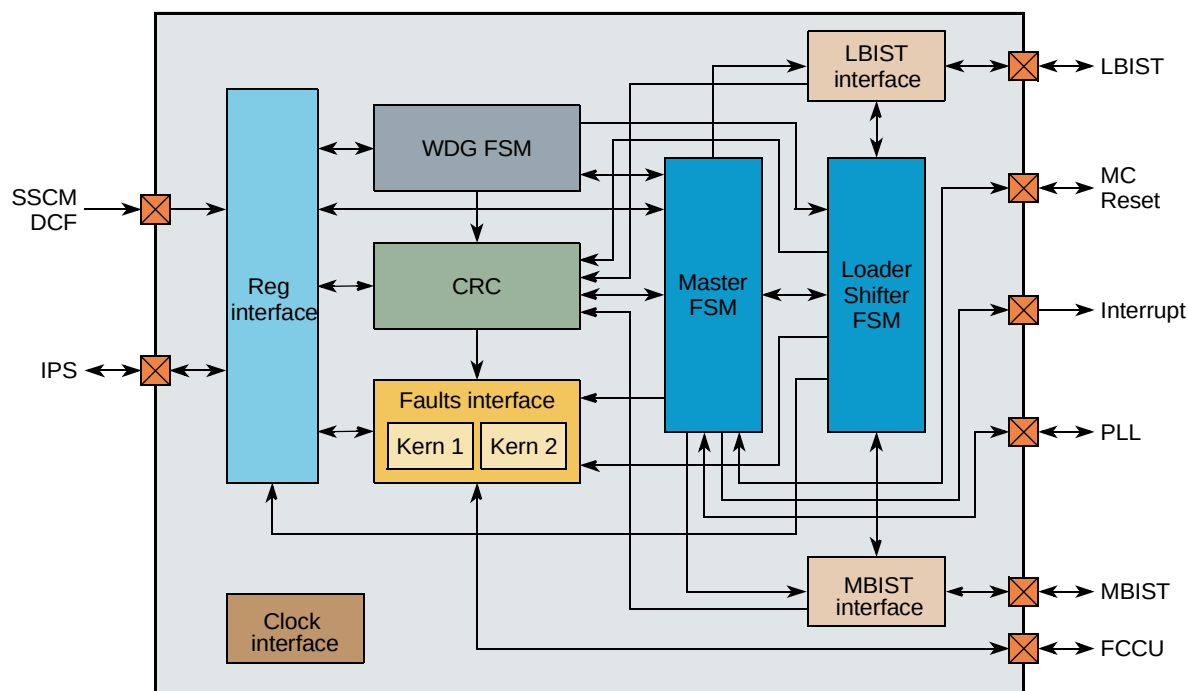


Figure 2046. STCU2 block diagram

The STCU2 module includes the following sub-modules:

- **Reg interface:** This module includes the registers, the security key logic, the IPS and SSCM DCF bus interface
- **CRC:** This module is the core of the Self-Checking architecture of the STCU2. It samples a set of selected internal signals when STCU2 is running.
- **Fault interface:** This module collects the error conditions related to L/MBIST execution and STCU2 internal failures and, depending on the UF/RF configuration of each one of them, sets the global STCU2 UF or RF flag. It also manages the UF/RF fault lines and the Set/Clear injection mechanism provided by the FCCU. To improve the intrinsic safety of this critical logic, the generation logic is duplicated.
- **Clock Block:** This module manages the internal and the L/MBIST TCK clock prescaler, the internal Clock Gating Power Saving and the wake-up clock feature.
- **WDG FSM:** This module includes two different Watchdogs. The first one is Hard Coded and is used to Auto Lock the STCU2 access forcing a reset condition on the Double Security Key registers while the second one has a double functionality. After a reset event initializes the STCU2, it is used as Hard Coded Watchdog time-out while during

L/MBIST run it is used as programmable Watchdog timer to check the L/MBIST have been completed in the assigned to L/MBIST time slot.

- **STCU2 Master FSM:** This module includes the main FSM of the STCU2 used to coordinate and schedule all the operations performed during the Self-Test Sequence.
- **STCU2 Load Shifter FSM:** This module includes the common shifter register and the related state machine used to program the L/MBIST registers and read back the data to be checked at the end of each Test operation.
- **LBIST interface:** This module includes the interface between the device's LBIST engines and the STCU2 controllers.
- **MBIST interface:** This module includes the interface between the MBIST controller and STCU2 controllers.

82.3.1 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers
- Byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode only in the following scenario:

- The Self-Test sequence parameters are loaded before Off-Line Self-test is executed
- The IPS access on some specific register (see [Section 82.5, Register description](#)) is allowed by software only when the bit STCU_CFG.WRP is cleared
- In all the other conditions the write operations are not supported and all the STCU2 registers can be only read. Only exceptions are the following rw bits: STCU_CFG.WRP, STCU_ERR_STAT.UFSF and STCU_ERR_STAT.RFSF. See related register description for additional details

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/hword/word (free byte enables or other operations) on each register
- Any write operation on Double Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Double Security keys have not been applied
- Any write operation performed on registers when the STCU_CFG.WRP bit is set and the access is performed through software (IPS interface)
- Any write operation performed through software (IPS interface) on row registers
- Any write operation performed off-line on registers in which writing operation is allowed only in the On-Line Self-Test phase when STCU_CFG[WRP] = 1
- Any write operation performed on Read Only registers

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

In case there are write operations on bits marked as reserved, the transfer error is not generated.

For more details about the registers protection features, see [Section 82.5, Register description](#).

82.4 Functional description

82.4.1 FSM description

The module has three state machines that work together: Master State Machine, Loader/Shifter State Machine, and the Watchdog State Machine. Basically the Master State Machine is the core unit of the STCU2 module. It coordinates all the Self Test operations and the other State Machines. The Loader/Shifter State Machine is used to program the MBIST and the LBIST parameters and to retrieve the related results depending on the parameters stored into the STCU2 registers and under the control of the Master State Machine. The Watchdog State Machine evaluates all the schedule time for MBIST and LBIST and the time-out in case of wrong STCU2 programming.

82.4.2 Reset management

The asynchronous reset line initializes the STCU2 module status forcing the Off-Line Self-Test condition and cleaning up all the registers and state machine of the module.

The reset signals generated by the STCU2 module at the end of the Self-Test execution phase depends on the Self-Test conditions and on the content of the STCU_LBRMSW register as reported in the following.

82.4.2.1 Off-Line reset generation

At the end of the Off-Line Self-Test execution the STCU2 raises the global functional reset.

82.4.2.2 On-Line reset generation (only MBIST)

The On-Line execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is because the software application has to manage all the possible consequences related to the write operations performed by the MBIST execution other than prevent any read/write operation during the Self-Test execution. If necessary, the software must also restore the memory content (only for RAMs) at the end of the Self-Test execution.

82.4.2.3 On-Line Reset generation (LBIST enabled)

The On-Line execution of the LBIST generates at the end of the execution two possible types of reset depending on the status of the STCU_LBRMSW register. See [Section 82.5.17, STCU2 On-Line LBIST Reset Management \(STCU_LBRMSW\)](#).

In case all the selected LBIST do not require the global functional reset, only the LBISTed partitions are cleaned up applying a pulse on the dedicated reset line while when at least one of the selected LBIST requires the global functional reset, then the global functional reset will be raised at the end of the On-Line Self-Test execution.

Note: In case there is an ABORT condition detected by STCU2 during the On-Line Self-Test execution, independently of the value of the STCU_LBRMSW register, only the related reset line is pulsed to prevent any potential dangerous system level operations.

82.4.3 LBIST and MBIST scheduling

The STCU2 module has been designed to be very flexible allowing to program the parallel/serial execution of the MBIST or LBIST depending on the power/timing/coverage constraints. The limitation in the programming flexibility is described in [Section 82.7.4, Design implementation information](#).

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU_CFG.PTR. The first LBIST is mapped on 00h, the second on 01h, and so on. The first MBIST is mapped at the address 10h, the second on 11h and so on. The additional pointers are in the bitfield STCU_LB_CTRL[X].PTR for LBIST[X] (where X is the selected LBIST) and STCU_MB_CTRL[Y].PTR for MBIST (where Y is the selected MBIST) and have to be filled depending on the selected sequence of run. The additional bitfield STCU_LB_CTRL.CSM and STCU_MB_CTRL.CSM provide the flexibility to run concurrently or sequentially the chosen set of the LBIST or MBIST or to close the linked list setting the NIL pointer. For more details, see [Section 82.5.34, STCU2 LBIST Control Register \(STCU_LB_CTRL\)](#) and [Section 82.5.46, STCU2 MBIST Control Register \(STCU_MB_CTRL\)](#).

82.4.4 ABORT management

The STCU2 module provides two mechanisms to perform the On-Line Self-Test execution abort. When the abort is detected in both cases, the On-Line Self-Test operation stops running and the status of the currently running MBIST or LBIST is saved into the related registers to provide intermediate results that might be useful in case of debug or in case of concurrent run. In case of LBIST run, see [Section 82.4.2, Reset management](#), for more details about reset specific management when abort is detected. The following sections describe the differences between these two modalities.

82.4.4.1 Hardware ABORT management

The STCU2 enters in this abort condition as a consequence of a Functional reset that requires to stop the Self-Test execution.

When hardware abort is detected the bit ABORTHW of the STCU_ERR_STAT register is set. This flag allows the software to diagnose what has happened during the On-Line Self-Test execution run.

82.4.4.2 Software ABORT management

The STCU2 enters in the abort condition because the software set the bit RUNSW_ABORT in the STCU_RUNSW register. In order to allow the software to understand when STCU2 has completed the abort sequence, this bit will be maintained active until the abort sequence is completed. At that time the STCU2 will clear this bit as described in [Section 82.5.3, STCU2 Run Software Register \(STCU_RUNSW\)](#).

When software abort is raised the bit ABORTSW of the STCU_ERR_STAT register is set. Even if the abort condition is forced by software operation, this flag allows to diagnose what has happened during the On-Line Self-Test execution run.

82.4.5 PLL interface

The STCU2 module is able to directly control the PLL during the Off-Line Self-Test operations depending on the status of the STCU_RUN[MBPLEN] and STCU_RUN[LBPLEN] as described in [Section 82.5.3, STCU2 Run Software Register \(STCU_RUNSW\)](#).

When STCU_RUN[RUN] is set to start the Off-Line Self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configurations parameters according to the values programmed into the STCU_PLL_CFG register.

When the STCU_RUN[MBPLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock. As soon as the lock happens, the STCU forces the clock to switch to the PLL output clock. This allows the MBIST to proceed at the PLL output frequency. At the end of the MBIST run the clock source is reverted to the original one. Same behavior will be applied when STCU_RUN[LBPLEN] is active and LBIST is currently selected.

82.4.6 Interrupt interface

There are two interrupt sources:

- One triggered by MBIST (enabled by programming STCU_RUNSW[MBIE] = 1)
- One triggered by LBIST (enabled by programming STCU_RUNSW[LBIE] = 1)

As described in [Section 82.5.3, STCU2 Run Software Register \(STCU_RUNSW\)](#), register description, the interrupts are generated at the end of any concurrent run. This means that in case of multiple sequential run in the same session, the number of interrupts generated is equivalent to the number of the sequential session providing in such a way a fine granularity with respect the completed run.

The bits LBIFLG and MBIFLG in the [STCU2 Interrupt Flag Register \(STCU_INT_FLG\)](#) register flag these interrupt conditions and allow the clearing of interrupt requests. These two interrupt bits have to be cleared writing the related bit to 1 to enable the next interrupt request.

82.4.7 FCCU interface

The FCCU interface is the hardware flag mechanism towards the system, to indicate the occurrence of an Unrecoverable and/or a Recoverable failure during the Self Test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. In this case, the FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set/clear mechanism works.

82.4.8 Watchdogs

The STCU2 implements three different watchdog ensure that operations are finished in time.

82.4.8.1 Initialization sequence

The STCU2 has two initialization operating modes: program the Self-Test sequence and run it (Safety mode) or bypass completely the Self-Test sequence setting the bit

STCU_CFG.BYP. In case there are faults during the initialization phase preventing the selection of one of these two operating modes, a hard-coded Watchdog time-out flags the wrong behavior. See [Section 82.5.7, STCU2 Watchdog Register Granularity \(STCU_WDG\)](#), for more details.

82.4.8.2 LBIST and MBIST scheduling

The LBIST and MBIST execution time has to be configured as described in [Section 82.5.7, STCU2 Watchdog Register Granularity \(STCU_WDG\)](#), to account for the overall execution time of the Self-Test sequence. In case the selected LBISTs or MBISTs are not yet completed during the Off-line Self-Test assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged into STCU_ERR_STAT.WDTO and STCU_MBEL/M/H or STCU_LBE registers while in case of On-Line Self-Test into STCU_ERR_STAT.WDTOSW and STCU_MBELSW/MSW/HSW or STCU_LBESW.

In case of multiple sequential run in the same On-Line/Off-Line session and the time-out happens in the middle of a sequential run, the next sequential run will be skipped and the execution ends with the current updated status of the registers reported above.

In all the cases, the STCU2 reset generation is managed in one of the modes described in [Section 82.4.2, Reset management](#).

82.4.8.3 Write access time-out

The access on the STCU2 registers during the Self Test configuration phase (both On-Line/Off-Line) is protected by a key mechanism to prevent any unwanted access, as described in the STCU_SKC register description. In addition to this safety feature, there is a hardware Watchdog that, after 1024 STCU2 clock cycles, lock the access requiring to apply again the STCU_SKC keys.

This feature is particularly useful in case STCU_CFG[WRP] = 0 during the software Self Test configuration since in this case the software application might enable the write access to the STCU2 registers.

82.4.9 CRC

The CRC block is used to increase the intrinsic coverage of the STCU2 module and implements a 32-bit ethernet protocol polynomial to evaluate the signature of the most important and critical internal signals. This feature is by default switched off but it can be turned on setting the bit STCU_CFG.CRCEN. Once this bit is set the expected signature has to be programmed into the STCU_CRCE register; at the end of the Self-Test execution the STCU2 update the content of the STCU_CRCR register with the evaluated signature and compare this value with the expected one previously programmed into STCU_CRCE. In case of mismatches a UF/RF condition is flagged depending on the fault mapping selected into the STCU_ERR_FM register.

82.4.10 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without the CPU intervention after a reset trigger event initializes the STCU2. This bus interface has the priority over the IPS.

82.5 Register description

The STCU2 registers are listed in [Table 1906](#).

Table 1906. STCU2 register set

Address offset	Register Name	Location
0000h	STCU2 Run Register (STCU_RUN)	on page 3542
0004h	STCU2 Run Software Register (STCU_RUNSW)	on page 3543
0008h	STCU2 SK Code Register (STCU_SKC)	on page 3544
000Ch	STCU2 Configuration Register (STCU_CFG)	on page 3546
0010h	STCU2 PLL Configuration Register (STCU_PLL_CFG)	on page 3548
0014h	STCU2 Watchdog Register Granularity (STCU_WDG)	on page 3549
0018h	STCU2 Interrupt Flag Register (STCU_INT_FLG)	on page 3550
001Ch	STCU2 CRC Expected Status Register (STCU_CRCE)	on page 3551
0020h	STCU2 CRC Read Status Register (STCU_CRCR)	on page 3552
0024h	STCU2 Error Register (STCU_ERR_STAT)	on page 3552
0028h	STCU2 Error FM Register (STCU_ERR_FM)	on page 3555
002Ch	STCU2 Off-Line LBIST Status Register (STCU_LBS)	on page 3556
0030h	STCU2 Off-Line LBIST End Flag Register (STCU_LBE)	on page 3557
0034h	STCU2 On-Line LBIST Status Register (STCU_LBSSW)	on page 3558
0038h	STCU2 On-Line LBIST End Flag Register (STCU_LBESW)	on page 3559
003Ch	STCU2 On-Line LBIST Reset Management (STCU_LBRMSW)	on page 3559
0040h	STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)	on page 3560
0044h	STCU2 Off-Line MBIST Status Low Register (STCU_MBSL)	on page 3561
0048h	STCU2 Off-Line MBIST Status Medium Register (STCU_MBSM)	on page 3562
004Ch	STCU2 Off-Line MBIST Status High Register (STCU_MBSH)	on page 3562
0050h	STCU2 Off-Line MBIST End Flag Low Register (STCU_MBEL)	on page 3563
0054h	STCU2 Off-Line MBIST End Flag Medium Register (STCU_MBEM)	on page 3564
0058h	STCU2 Off-Line MBIST End Flag High Register (STCU_MBEH)	on page 3565
005Ch	STCU2 On-Line MBIST Status Low Register (STCU_MBSLSW)	on page 3565
0060h	STCU2 On-Line MBIST Status Medium Register (STCU_MBSMSW)	on page 3566
0064h	STCU2 On-Line MBIST Status High Register (STCU_MBSHSW)	on page 3567
0068h	STCU2 On-Line MBIST End Flag Low Register (STCU_MBELSW)	on page 3568
006Ch	STCU2 On-Line MBIST End Flag Medium Register (STCU_MBEMSW)	on page 3569
0070h	STCU2 On-Line MBIST End Flag High Register (STCU_MBEHSW)	on page 3570
0074h	STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)	on page 3571
0078h	STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)	on page 3572
007Ch	STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)	on page 3573

Table 1906. STCU2 register set (continued)

Address offset	Register Name	Location
0100h+(NLBIST-1)*40h	STCU2 LBIST Control Register (STCU_LB_CTRL)	on page 3574
0104h+(NLBIST-1)*40h	STCU2 LBIST PC Stop Register (STCU_LB_PCS)	on page 3576
0108h+(NLBIST-1)*40h	STCU2 LBIST PRPG Low Register (STCU_LB_PRPGL)	on page 3577
010Ch+(NLBIST-1)*40h	STCU2 LBIST PRPG High Register (STCU_LB_PRPGH)	on page 3578
0110h+(NLBIST-1)*40h	STCU2 Off-Line LBIST MISR Expected Low Register (STCU_LB_MISREL)	on page 3579
0114h+(NLBIST-1)*40h	STCU2 Off-Line LBIST MISR Expected High Register (STCU_LB_MISREH)	on page 3580
0118h+(NLBIST-1)*40h	STCU2 Off-Line LBIST MISR Read Low Register (STCU_LB_MISRRL)	on page 3581
011Ch+(NLBIST-1)*40h	STCU2 Off-Line LBIST MISR Read High Register (STCU_LB_MISRRLH)	on page 3581
0120h+(NLBIST-1)*40h	STCU2 On-Line LBIST MISR Expected Low Register (STCU_LB_MISRELSW)	on page 3582
0124h+(NLBIST-1)*40h	STCU2 On-Line LBIST MISR Expected High Register (STCU_LB_MISREHSW)	on page 3583
0128h+(NLBIST-1)*40h	STCU2 On-Line LBIST MISR Read Low Register (STCU_LB_MISRRLSW)	on page 3584
012Ch+(NLBIST-1)*40h	STCU2 On-Line LBIST MISR Read High Register (STCU_LB_MISRRLHSW)	on page 3585
0600h+(NMCUT-1)*4h	STCU2 MBIST Control Register (STCU_MB_CTRL)	on page 3586

82.5.1 Register description conventions

The register information is presented in:

- Register figures
- Field-description tables
- Associated text

The register figures show the field structure using the conventions in [Figure 2047](#).

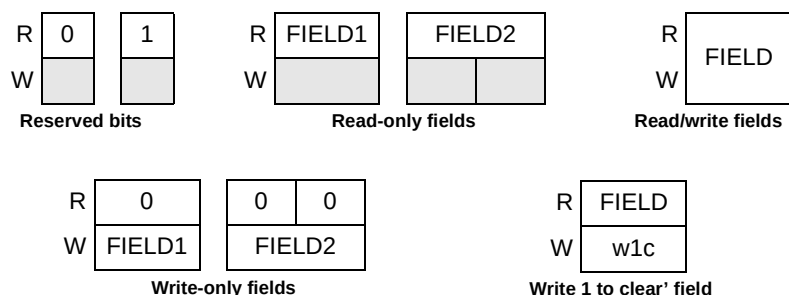


Figure 2047. Register figure conventions

The numbering of register bits and fields is as follows:

- Register bit numbers, shown at the top of each figure, use the bit ordering (0, 1, 2, ..., 31).
- Multi-bitfields within a register use the bit ordering (... , 2, 1, 0).

82.5.2 STCU2 Run Register (STCU_RUN)

The STCU_RUN register defines the RUN bit to start the off-line self testing procedure as long as some configuration bits required ONLY during this Self-test phase.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Figure 2048. STCU2 Run Register (STCU_RUN)

Offset: 0000h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0				0	0	0	0	0	0	0	
W						BYP	MBPLEN	LBPLEN								RUN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1907. STCU_RUN field descriptions

Field	Description
BYP	Bypass mode 0: Off-Line Self-Test is active. In case the RUN bit into the STCU_RUN register is not set before the Hard-Coded WDG Time-out, the WDTO error is generated into the STCU_ERR_STAT register. 1: Off-Line Self-Test is completely bypassed and the access to STCU2 is locked until the STCU_SCK register is written according to the request access mode.
MBPLEN	Off-Line MBIST with PLL Enabled 0: Off-Line MBIST is executed without using the on-chip PLL. 1: Off-Line MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU_PLL_CFG register.

Table 1907. STCU_RUN field descriptions (continued)

Field	Description
LBPLEN	Off-Line LBIST with PLL Enabled 0: Off-Line LBIST is executed without using the on-chip PLL. 1: Off-Line LBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU_PLL_CFG register.
RUN	RUN: RUN 0: Idle 1: Off-Line Self testing procedure is running The RUN bit is automatically cleared by the STCU2 when the off-line self testing procedure has been completed.

82.5.3 STCU2 Run Software Register (STCU_RUNSW)

The STCU_RUNSW register defines the RUN bit to start the on-line self testing procedure as long as some configuration bits required ONLY during this Self-test phase.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU_CFG[WRP] = 0
- On-line Self-Test phase is active. The On-Line condition starts when the Off-Line is over. In this condition, system is alive and the SW can program the STCU

Figure 2049. STCU2 Run Software Register (STCU_RUNSW)

Offset: 0004h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0					0	0	0	0	0	0		
W					MBIE	LBIE	MBSWPLEN	LBSWPLEN							RUNSW_ABORT	RUNSW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1908. STCU_RUNSW field descriptions

Field	Description
MBIE	MBIST Interrupt Enable 0: Interrupt is not generated at the end of the software MBIST execution phase of the selected Memories. 1: At the end of the software MBIST execution phase of the concurrent selected memories the STCU2 forces an interrupt request to notify MBISTs of that concurrent selected memories are over. In case there is a sequence of concurrent run in the same Self-Test run, the interrupt request is generated at the end of each concurrent run provided that the software will clear the STCU_INT_FLG[MBIFG] bit in between.
LBIE	LBIST Interrupt Enable 0: Interrupt is not generated at the end of the software LBIST execution phase of the selected LBIST. 1: At the end of the software LBIST execution phase of the concurrent selected LBIST the STCU2 forces an interrupt request to notify LBIST of the concurrent selected partitions are over. In case there is a sequence of concurrent run in the same Self-Test run, the interrupt request is generated at the end of each concurrent run provided that the software will clear the STCU_INT_FLG.LBIFG bit in between.
MBSWPLEN	On-Line MBIST with PLL Enabled 0: On-Line MBIST is executed without using the on-chip PLL. 1: On-Line MBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
LBSWPLEN	On-Line LBIST with PLL Enabled 0: On-Line LBIST is executed without using the on-chip PLL. 1: On-Line LBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
RUNSW_ABORT	Software abort of the self-test currently running This bit is automatically cleared by the STCU2 when the software self-test abort sequence has been completed. 0: On-line self-test abort is not requested. 1: On-line self-test abort is requested.
RUNSW	RUNSW 0: Idle 1: On-line self testing procedure is running The RUNSW bit is automatically cleared by the STCU2 when the on-line self testing procedure has been completed.

82.5.4 STCU2 SK Code Register (STCU_SKC)

The STCU_SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers. In order to unlock the STCU2 access after:

- The STCU2 asynchronous reset
- The end of the STCU2 run

the software (IPS bus) or the SSCM interfaces have to apply the following sequence:

- write the key1 into the STCU_SKC register
- write the key2 into the STCU_SKC register

Depending on the off/on-line test step, the two keys will be different. Byte write operation is not allowed since the full key has to be recognized as one unit.

It has to be noticed that after the Self-Test sequence has been completed or the Bypass feature has been enabled (setting the bit BYP into the STCU_CFG), the SSCM interface is no more available.

In case of invalid access or sequence (Key1/2 have to be applied consecutively), a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

In case the STCU2 register access last more cycles than the one defined into the Hard-coded WDG time-out, the STCU2 write access is locked and the WDG and Register ITF clocks are switched off. Also in this case, in order to enable again the write access to the STCU2 and the WDG and Register ITF clocks, it is required to apply again the sequence.

In case it is required to extend the STCU2 register access cycles before the Hard-coded WDG time-out expires, only the Key2 has to be applied. The effect of this write operation is to re-initialize the WDG time-out counter. In such a condition, Key1 has not to be applied otherwise a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

The STCU_SKC register is not readable. The value 00000000h is always returned in case of read operation.

Figure 2050. STCU2 SK Code Register (STCU_SKC)

Offset: 0008h

Access: User write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SKC[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SKC[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1909. STCU_SKC field descriptions

Field	Description
SKC	STCU2 security key code for off-line Test = D3FEA98Bh: Key1 to unlock the write access the STCU2 = 2C015674h: Key2 to unlock the write access the STCU2
	STCU2 security key code for on-line Test = 753F924Eh: Key1 to unlock the write access the STCU2 = 8AC06DB1h: Key2 to unlock the write access the STCU2

82.5.5 STCU2 Configuration Register (STCU_CFG)

The STCU_CFG register includes the global configuration of the STCU2 and can be updated both in the Off/On-Line Test steps.

The access to this register is described in [Figure 2051](#). It further depends on the state of the WRP bit as follows:

- When WRP = 0: The register can be written during the on-line Self-Test case without restrictions
- When WRP = 1:
 - The only bit that can be written without any restriction is WRP.
 - In case there are software operations that write all the register's bits, a transfer error is raised only if the value of the selected byte written differs from the current status of the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

Figure 2051. STCU2 Configuration Register (STCU_CFG)

Offset: 000Ch								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PTR ⁽¹⁾							LB_DELAY ⁽¹⁾							
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	WRP	0	0	CRCEN ⁽¹⁾	PMOSEN ⁽¹⁾	MBU ⁽¹⁾	CLK_CFG ⁽¹⁾		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.

Table 1910. STCU_CFG field descriptions

Field	Description
PTR	LBIST or MBIST pointer PTR defines the logical pointer to the first LBIST or MBIST to be scheduled when the self testing procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. If PTR = 0 then the LBIST0 is initially scheduled. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1): pointer to MBIST 7Fh: pointer to NIL. No BIST execution.
LB_DELAY	Delay LBIST run LB_DELAY defines the delay between the LBIST starts when more than a single LBIST is selected of being executed concurrently with the purpose of smoothing the power consumption transient. The allowed delay time are the following: 00h: No delay 01h: 1 × 16 STCU2 Core clock cycles 02h: 2 × 16 STCU2 Core clock cycles 03h: 3 × 16 STCU2 Core clock cycles ... FDh: 253 × 16 STCU2 Core clock cycles FEh: 254 × 16 STCU2 Core clock cycles FFh: 255 × 16 STCU2 Core clock cycles Note that in case the Frequency of the STCU2 core clock is 16 MHz, the delay parameters is expressed in μs.
WRP	Write Protection 0: Specific STCU2 registers can be written through IPS bus interface after Off-Line Self-Test sequence has been completed 1: Specific STCU2 registers cannot be written though IPS also after Off-Line Self-Test sequence has been completed preventing in such a way any user application write operation
CRCEN	CRCEN: <i>CRC Enable comparison</i> 0: The CRC comparison is not performed and the STCU_ERR_STAT.CRCS[SW] and STCU_ERR_STAT.UF/RF global flags are not updated in case of mismatches between STCU2.CRCE and STCU2.CRCR values 1: The CRC comparison is performed and the status is updated into the related flags of the STCU_ERR_STAT register
PMOSEN	MBIST PMOS Test Enable 0: MBIST PMOS Test is not enabled 1: MBIST PMOS Test is enabled

Table 1910. STCU_CFG field descriptions (continued)

Field	Description
MBU	<i>MBIST MBU Test Enabled</i> 0: MBIST Full algo or reduced one without PMOS test (PMOSEN set to 0) is applied. 1: MBIST simplified Multi Bit Upset algo is used to check the RAM. In case a ROM is selected, the applied algo will be Standard one without PMOS Test. <i>This bit overwrite the PMOSEN feature.</i>
CLK_CFG	Logic, Memory Bist and STCU2 core CLK <i>Clock configuration</i> CLK_CFG defines the ratio between the sys_clk and the TCK used to program both the LBIST and the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are the following: 000: sys_clk 001: sys_clk/2 010: sys_clk/3 ... 101: sys_clk/6 110: sys_clk/7 111: sys_clk/8

82.5.6 STCU2 PLL Configuration Register (STCU_PLL_CFG)

The STCU_PLL_CFG register defines the parameters used to program the PLL only when the Off-line L/MBIST are performed and STCU_RUN[MBPLEN] = 1 or STCU_RUN[LBPLEN] = 1.

In the On-line condition, these bits are not effective. It is also possible to set the PLL during active On-line mode to prevent any potential functionality issue.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Figure 2052. STCU2 PLL Configuration Register (STCU_PLL_CFG)

Offset: 0010h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PLLODF						0	0	0	0	0	PLLIDF		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	PLLLDF						
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Table 1911. STCU_PLL_CFG field descriptions

Field	Description
PLLODF	PLL Output Division Factor The value of this field drives the Output Division factor of the PLL embedded into the device. Refers to the PLL specifications to define the exact mapping between the PLLODF value and the related PLL's Output Division factor.
PLLIDF	PLL Input Division Factor The value of this field drives the Input Division factor of the PLL embedded into the device. Refers to the PLL specifications to define the exact mapping between the PLLIDF value and the related PLL's Input Division factor.
PLLLDF	PLL Loop Division Factor The value of this field drives the Loop Division factor of the PLL embedded into the device. Refers to the PLL specifications to define the exact mapping between the PLLLDF value and the related PLL's Loop Division factor.

82.5.7 STCU2 Watchdog Register Granularity (STCU_WDG)

The STCU_WDG register defines the time budget of LBIST and MBIST execution providing a protection mechanism in case of dead-lock or end-less conditions during the Self-Test procedure. When the Off-line Self-Test sequence is not run and the BYP bit of the STCU_CFG register is not set, the bits 15...0 defines the time-out before the bit WDTO of the STCU_ERR register is set and the STCU2 core clock is switched off.

In case Off/On-line Self Test sequence is run, it defines the time-out of the execution run.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2053. STCU2 Watchdog Register Granularity (STCU_WDG)

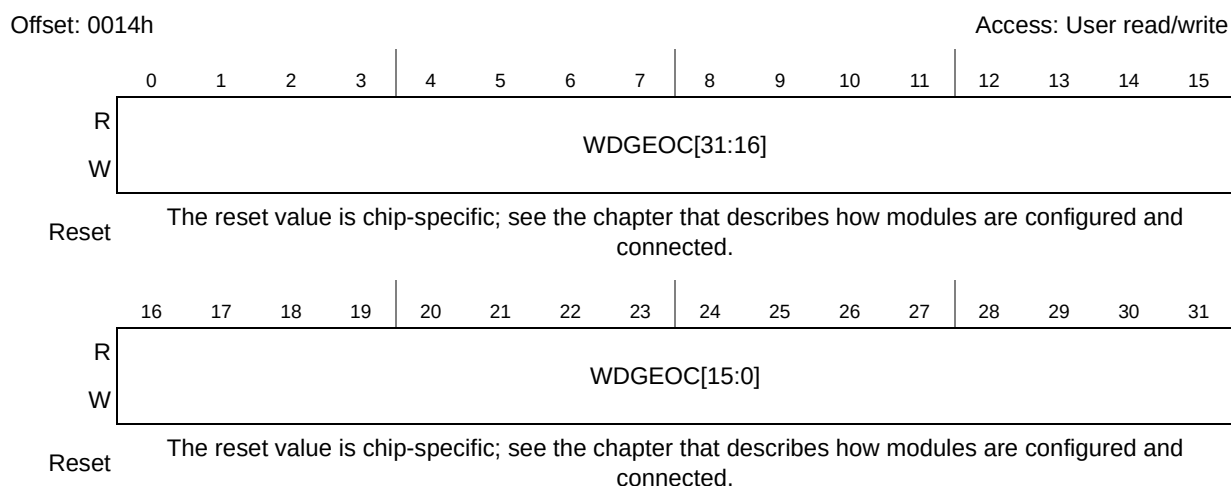


Table 1912. STCU_WDG field descriptions

Field	Description
WDGEOC	Watchdog End of Count Timer This value has to be set in order to define the time budget related to the Off/On-Line Self Test execution and check that everything is correctly working within this slot of time. The allowed delay time are the following: 0000 0000h: 1 × 16 STCU2 Core clock cycles 0000 0001h: 2 × 16 STCU2 Core clock cycles 0000 0002h: 3 × 16 STCU2 Core clock cycles ... FFFF FFFDh: 4294967294 × 16 STCU2 Core clock cycles FFFF FFFEh: 4294967295 × 16 STCU2 Core clock cycles FFFF FFFFh: 4294967296 × 16 STCU2 Core clock cycles

82.5.8 STCU2 Interrupt Flag Register (STCU_INT_FLG)

The STCU_INT_FLG register includes the MBIST and LBIST interrupt pending bits. These bits are effective only during the On-Line Self-Test phase and are managed only when the related control bit into the STCU_RUNSW register (MBIE and LBIE) are enabled.

Figure 2054. STCU2 Interrupt Flag Register (STCU_INT_FLG)

Offset: 0018h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MBIFLG	LBIFLG
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1913. STCU_INT_FLG field descriptions

Field	Description
MBIFLG	<i>MBIST Interrupt Flag</i> 0: No interrupt is pending 1: An interrupt, highlighting that the On-Line execution of the concurrent scheduled MBIST is completed, is pending. Write 1 to clear the bit.
LBIFLG	<i>LBIST Interrupt Flag</i> 0: No interrupt is pending 1: An interrupt, highlighting that the On-Line execution of the concurrent scheduled LBIST is completed, is pending. Write 1 to clear the bit.

82.5.9 STCU2 CRC Expected Status Register (STCU_CRCE)

The STCU_CRCE register includes the expected signature of the CRC-32 (ethernet protocol). The CRC-32 bit engine is initialized at STCU_CRCE_RES and computes the CRC signature of the STCU2's critical signals and when STCU_CFG[CRCE] = 1, it is able to provide the Self-Check capability of the STCU2 managing the CRCS[SW] bits of the STCU_ERR_STAT register. The polynomial CRC used to evaluate the status of this register is the following:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Equation 91 CRC-32 (ethernet protocol)

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2055. STCU2 CRC Expected Status Register (STCU_CRCE)

Offset: 001Ch

Access: User read/write

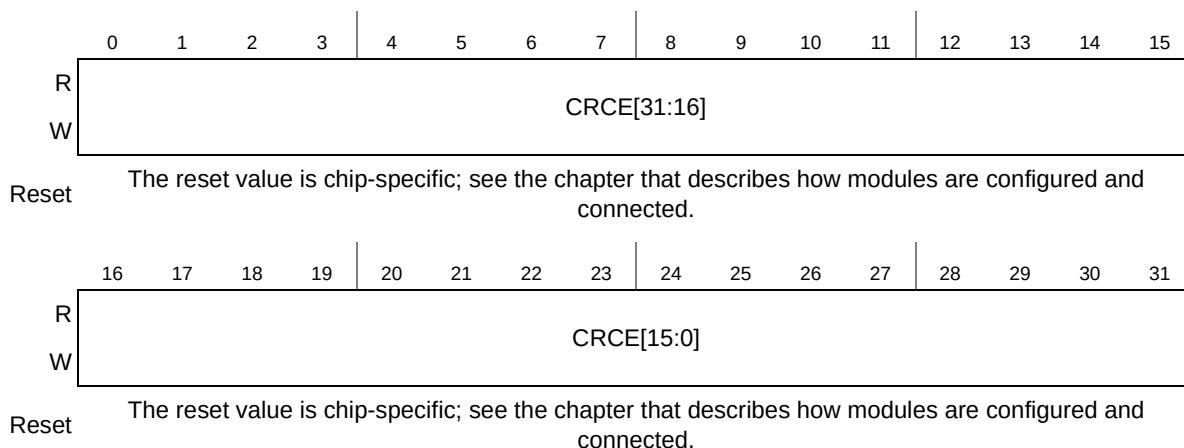


Table 1914. STCU_CRCE field descriptions

Field	Description
CRCE	CRC expected signature

82.5.10 STCU2 CRC Read Status Register (STCU_CRCR)

The STCU_CRCR register reports the value obtained at the end of the Off/On-line Self-Test sequence. It might be used for diagnoses and also as additional check with respect to the CRCS[SW] bit of the STCU_ERR_STAT register.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2056. STCU2 CRC Read Status Register (STCU_CRCR)

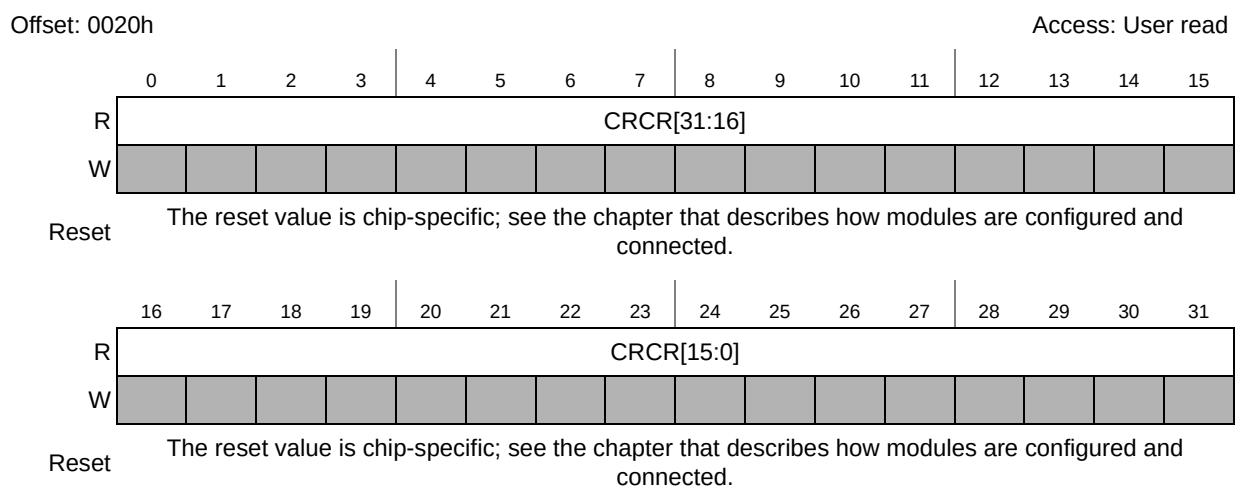


Table 1915. STCU_CRCR field descriptions

Field	Description
CRCR	Read CRC signature

82.5.11 STCU2 Error Register (STCU_ERR_STAT)

The STCU_ERR_STAT register includes the status flags related to the STCU2 internal error conditions occurred during the configuration or the On/Off-Line self testing execution.

The UFSF and RFSF can be set/cleared using the FCCU dedicated channels as described in [Section 82.7.4, Design implementation information](#).

The access to this register is described in [Figure 2057](#) and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality has been implemented to

prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set/clean capability.

Figure 2057. STCU2 Error Register (STCU_ERR_STAT)

Offset: 0024h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	ABORTHW ⁽¹⁾	ABORTSW ⁽¹⁾	0	0	0	LOCKESW ⁽¹⁾	WDTOSW ⁽¹⁾	CRCSSW ⁽¹⁾	ENGESW ⁽¹⁾	INVP ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	UFSF	RFSF	0	0	0	LOCKE	WDTO	CRC	ENGE	INVP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can always read this field. The content of this field is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Table 1916. STCU_ERR_STAT field descriptions

Field	Description
ABORTHW	<i>On-line hardware abort flag</i> 0: No hardware abort was requested during the On-Life Self-Test sequence 1: A hardware abort was detected during the On-Life Self-Test sequence
ABORTSW	<i>On-line software abort flag</i> 0: No software abort was requested during the On-Life Self-Test sequence 1: A software abort was detected during the On-Life Self-Test sequence
LOCKESW	<i>On-Line LOCK Error</i> 0: In case PLL is enabled, it is correctly locked during the Self-Test sequence 1: When the PLL is enabled, this flag highlight that there has been an unexpected PLL unlock event during the On-Line Self-Test sequence execution. The On-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU_RUNSW.LBSWPLEN is set and/or during the MBIST run when STCU_RUNSW[MBSWPLEN] = 1.
WDTOSW	<i>On-Line Watchdog time-out</i> 0: LBIST and MBIST time slot have been completed within the assigned watchdog time. 1: LBIST and MBIST time slot have not been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: – MBIST BEND status flags that at least one of the selected MBIST run is not finished – lbist_done of the selected LBIST flags the LBIST run is/are not finished

Table 1916. STCU_ERR_STAT field descriptions (continued)

Field	Description
CRCSSW	<i>On-Line CRC Status</i> This flag is activated ONLY when On-Line Self Test is performed and STCU_CFG.CRCEN is set to 1 otherwise it is always forced to 0 preventing in such a way the generation of the related STCU_ERR_STAT.UFSF/RFSF bits. In both the cases, the content of the STCU_CRCCR register report the current evaluated CRC. 0: Successful CRC comparison or comparison has been masked (STCU_CFG.CRCEN set to 0) 1: Failed CRC comparison
ENGESW	<i>On-Line Engine Error</i> 0: Valid Engine execution 1: Invalid Engine execution. The following conditions set this bit: – STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code – Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned – Serial bend status of the selected MBIST and the global BEND flag are not aligned
INVPSW	<i>On-Line Invalid Pointer</i> 0: Valid linked pointer list 1: Invalid linked pointer list. The following conditions set this bit: – Initial LBIST or MBIST pointer is out of range – LBIST is selected when MBIST is concurrently running or vice versa – Error in the LBIST/MBIST linking (execution generates an infinite loop)
UFSF	Unrecoverable Faults Status Flag This flag report the global status of the UF. 0: No errors that trigger the Unrecoverable Faults condition 1: There are errors that trigger the Unrecoverable Faults condition
RFSF	Recoverable Faults Status Flag 0: No errors that trigger the Recoverable Faults condition 1: There are errors that trigger the Recoverable Faults condition
LOCKE	<i>Off-Line LOCK Error</i> 0: The PLL is correctly locked during the Self-Test sequence 1: When the PLL is enabled, this flag highlight that there has been an unexpected PLL unlock event during the Off-Line Self-Test sequence execution. The On-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU_RUN[LBPLEN] = 1 and/or during the MBIST run when STCU_RUN[MBPLEN] = 1.
WDTO	<i>Off-Line Watchdog time-out</i> 0: LBIST and MBIST time slot have been completed within the assigned watchdog time. 1: LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: – The STCU_RUN[RUN] bit or the STCU_CFG[BYP] bit are not set before the Watchdog reach the End of Count – MBIST BEND status flags that at least one of the selected MBIST run is not finished – lbist_done of the selected LBIST flags the LBIST run is/are not finished

Table 1916. STCU_ERR_STAT field descriptions (continued)

Field	Description
CRCS	<i>Off-Line CRC status</i> This flag is activated ONLY when Off-Line Self Test is performed STCU_CFG[CRCEEN] is set to 1 otherwise it is always forced to 0 preventing in such a way the generation of the related STCU_ERR_STAT.UFSF/RFSF bits. In both the cases, the content of the STCU_CRCCR register report the current evaluated CRC. 0: Successful CRC comparison or comparison has been masked (STCU_CFG[CRCEEN] programmed to 0) 1: Failed CRC comparison
ENGE	<i>Off-Line Engine Error</i> 0: Valid Engine execution 1: Invalid Engine execution. The following conditions set this bit: – STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code – Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned – Serial bend status of the selected MBIST and the global BEND flag are not aligned
INVP	<i>Off-Line Invalid pointer</i> 0: Valid linked pointer list 1: Invalid linked pointer list. The following conditions set this bit: – Initial LBIST or MBIST pointer is out of range – LBIST is selected when MBIST is concurrently running or vice versa – Error in the LBIST/MBIST linking (execution generates an infinite loop)

82.5.12 STCU2 Error FM Register (STCU_ERR_FM)

The STCU_ERR_FM register defines the fault mapping of the STCU2 faults described into the register STCU_ERR_STAT in terms of Unrecoverable or Recoverable Fault. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2058. STCU2 Error FM Register (STCU_ERR_FM)

Offset: 0028h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LOCKEUFM	WDTOUFM	CRCSUFM	ENGEUFM	INVPUFM
W												LOCKEUFM	WDTOUFM	CRCSUFM	ENGEUFM	INVPUFM

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1917. STCU_ERR_FM field descriptions

Field	Description
LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0: Recoverable Fault Mapping 1: Unrecoverable Fault Mapping
WDTOUFM	WatchDog time-out Unrecoverable Fault Mapping 0: Recoverable Fault Mapping 1: Unrecoverable Fault Mapping
CRCSUFM	CRC Status Unrecoverable Fault Mapping 0: Recoverable Fault Mapping 1: Unrecoverable Fault Mapping
ENGEUFM	Engine Error Unrecoverable fault Mapping 0: Recoverable Fault Mapping 1: Unrecoverable Fault Mapping
INVPUFM	Invalid Pointer Unrecoverable Fault Mapping 0: Recoverable Fault Mapping 1: Unrecoverable Mapping

82.5.13 STCU2 Off-Line LBIST Status Register (STCU_LBS)

The STCU_LBS register includes the results corresponding to the execution of the **selected** Off-Line LBIST.

The size of the register depends on the number of LBIST.

Figure 2059. STCU2 Off-Line LBIST Status Register (STCU_LBS)

Offset: 002Ch

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBS15	LBS14	LBS13	LBS12	LBS11	LBS10	LBS9	LBS8	LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
		LBS14	LBS13	LBS12	LBS11	LBS10	LBS9	LBS8	LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
				LBS12	LBS11	LBS10	LBS9	LBS8	LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
					LBS11	LBS10	LBS9	LBS8	LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
W									LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1918. STCU_LBS field descriptions

Field	Description
LBSx	Off-Line status of the selected LBIST 0: Failed LBIST execution 1: Successful LBIST execution

82.5.14 STCU2 Off-Line LBIST End Flag Register (STCU_LBE)

The STCU_LBE register includes the End Flag related to the execution of the **selected** Off-Line LBIST.

The size of the register depends on the number of LBIST.

Figure 2060. STCU2 Off-Line LBIST End Flag Register (STCU_LBE)

Offset: 0030h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBE15	LBE14	LBE13	LBE12	LBE11	LBE10	LBE9	LBE8	LBE7	LBE6	LBE5	LBE4	LBE3	LBE2	LBE1	LBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1919. STCU_LBE field descriptions

Field	Description
LBE _x	Off-Line End status of the selected LBIST 0: LBIST execution not yet completed 1: LBIST execution finished

82.5.15 STCU2 On-Line LBIST Status Register (STCU_LBSSW)

The STCU_LBSSW register includes the results corresponding to the execution of the **selected** On-Line LBIST.

The size of the register depends on the number of LBIST.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2061. STCU2 On-Line LBIST Status Register (STCU_LBSSW)

Offset: 0034h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBSSW15	LBSSW14	LBSSW13	LBSSW12	LBSSW11	LBSSW10	LBSSW9	LBSSW8	LBSSW7	LBSSW6	LBSSW5	LBSSW4	LBSSW3	LBSSW2	LBSSW1	LBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1920. STCU_LBSSW field descriptions

Field	Description
LBSSWx	LBSSWx: <i>On-Line status of the selected LBIST</i> 0: Failed LBIST execution 1: Successful LBIST execution

82.5.16 STCU2 On-Line LBIST End Flag Register (STCU_LBESW)

The STCU_LBESW register includes the End Flag related to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2062. STCU2 On-Line LBIST End Flag Register (STCU_LBESW)

Offset: 0038h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBESW15	LBESW14	LBESW13	LBESW12	LBESW11	LBESW10	LBESW9	LBESW8	LBESW7	LBESW6	LBESW5	LBESW4	LBESW3	LBESW2	LBESW1	LBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1921. STCU_LBESW field descriptions

Field	Description
LBESWx	LBESWx: <i>On-Line LBIST End status</i> 0: LBIST execution not yet completed 1: LBIST execution finished

82.5.17 STCU2 On-Line LBIST Reset Management (STCU_LBRMSW)

The STCU_LBRMSW register defines the On-Line reset mapping for each LBIST. Global or specific functional reset can be programmed depending on the On-Line Self-Test requirements and top level integration architecture.

The size of the register depends on the number of LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2063. STCU2 On-Line LBIST Reset Management (STCU_LBRMSW)

Offset: 003Ch

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBRMSW15	LBRMSW14	LBRMSW13	LBRMSW12	LBRMSW11	LBRMSW10	LBRMSW9	LBRMSW8	LBRMSW7	LBRMSW6	LBRMSW5	LBRMSW4	LBRMSW3	LBRMSW2	LBRMSW1	LBRMSW0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1922. STCU_LBRMSW field descriptions

Field	Description
LBRMSWx	On-Line <i>LBIST</i> Reset Management 0: Dedicated functional reset is pulsed at the end of <i>LBIST</i> run 1: Global functional reset is pulsed at the end of <i>LBIST</i> run <i>Note:</i> In case one of the selected <i>LBIST</i> has this bit set to '1', then only the Global functional reset will be pulsed.

82.5.18 STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)

The STCU_LBUFM register defines the fault mapping of each LBIST in terms of Unrecoverable or Recoverable Fault.

The size of the register depends on the number of LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2064. STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)

Offset: 0040h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBUFM15	LBUFM14	LBUFM13	LBUFM12	LBUFM11	LBUFM10	LBUFM9	LBUFM8	LBUFM7	LBUFM6	LBUFM5	LBUFM4	LBUFM3	LBUFM2	LBUFM1	LBUFM0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1923. STCU_LBUFM field descriptions

Field	Description
LBUFMx	<i>LBIST Unrecoverable Fault Mapping</i> 0: Recoverable Fault mapping 1: Unrecoverable Fault mapping

82.5.19 STCU2 Off-Line MBIST Status Low Register (STCU_MBSL)

The STCU_MBSL register includes the results corresponding to the execution of the **selected** Off-Line MBIST in the range NMCUT = 0..31.

The size of the register depends on the number of BISTed RAMs/ROMs.

Figure 2065. STCU2 Off-Line MBIST Status Low Register (STCU_MBSL)

Offset: 0044h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS31	MBS30	MBS29	MBS28	MBS27	MBS26	MBS25	MBS24	MBS23	MBS22	MBS21	MBS20	MBS19	MBS18	MBS17	MBS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS15	MBS14	MBS13	MBS12	MBS11	MBS10	MBS9	MBS8	MBS7	MBS6	MBS5	MBS4	MBS3	MBS2	MBS1	MBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1924. STCU_MBSL field descriptions

Field	Description
MBSx	<i>Off-Line status (NMCUT range = 0 to 31) of the selected MBIST</i> 0: Failed NMCUT BIST execution 1: No Fault detected during the NMCUT BIST execution <i>Note: This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished.</i>

82.5.20 STCU2 Off-Line MBIST Status Medium Register (STCU_MBSM)

The STCU_MBSM register includes the results corresponding to the execution of the **selected** Off-Line MBIST in the range NMCUT = 32...63.

The size of the register depends on the number of BISTed RAMs/ROMs.

Figure 2066. STCU2 Off-Line MBIST Status Medium Register (STCU_MBSM)

Offset: 0048h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS63	MBS62	MBS61	MBS60	MBS59	MBS58	MBS57	MBS56	MBS55	MBS54	MBS53	MBS52	MBS51	MBS50	MBS49	MBS48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS47	MBS46	MBS45	MBS44	MBS43	MBS42	MBS41	MBS40	MBS39	MBS38	MBS37	MBS36	MBS35	MBS34	MBS33	MBS32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1925. STCU_MBSM field descriptions

Field	Description
MBSx	<i>MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST</i> 0: Failed NMCUT BIST execution 1: No Fault detected during the NMCUT BIST execution <i>Note: This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished).</i>

82.5.21 STCU2 Off-Line MBIST Status High Register (STCU_MBSH)

The STCU_MBSH register includes the results corresponding to the execution of the **selected** Off-Line MBIST in the range NMCUT = 64...95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Figure 2067. STCU2 Off-Line MBIST Status High Register (STCU_MBSH)

Offset: 004Ch

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS95	MBS94	MBS93	MBS92	MBS91	MBS90	MBS89	MBS88	MBS87	MBS86	MBS85	MBS84	MBS83	MBS82	MBS81	MBS80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS79	MBS78	MBS77	MBS76	MBS75	MBS74	MBS73	MBS72	MBS71	MBS70	MBS69	MBS68	MBS67	MBS66	MBS65	MBS64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1926. STCU_MBSH field descriptions

Field	Description
MBSx	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST 0: Failed NMCUT BIST execution 1: No Fault detected during the NMCUT BIST execution Note: This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished).

82.5.22 STCU2 Off-Line MBIST End Flag Low Register (STCU_MBEL)

The STCU_MBEL register includes the End Flag related to the execution of the **selected** Off-Line MBIST in the range NMCUT = 0..31.

The size of the register depends on the number of BISTed RAMs/ROMs.

Figure 2068. STCU2 Off-Line MBIST End Flag Low Register (STCU_MBEL)

Offset: 0050h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31	MBE30	MBE29	MBE28	MBE27	MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1927. STCU_MBEL field descriptions

Field	Description
MBEx	Off-Line End status (0 to 31) of the selected MBIST 0: MBIST execution still ongoing 1: MBIST execution finished

82.5.23 STCU2 Off-Line MBIST End Flag Medium Register (STCU_MBEM)

The STCU_MBEM register includes the End Flag related to the execution of the **selected** Off-Line MBIST in the range NMCUT = 32..63.

The size of the register depends on the number of BISTed RAMs/ROMs.

Figure 2069. STCU2 Off-Line MBIST End Flag Medium Register (STCU_MBEM)

Offset: 0054h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE63	MBE62	MBE61	MBE60	MBE59	MBE58	MBE57	MBE56	MBE55	MBE54	MBE53	MBE52	MBE51	MBE50	MBE49	MBE48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE47	MBE46	MBE45	MBE44	MBE43	MBE42	MBE41	MBE40	MBE39	MBE38	MBE37	MBE36	MBE35	MBE34	MBE33	MBE32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1928. STCU_MBEM field descriptions

Field	Description
MBEx	Off-Line End status (32 to 63) of the selected MBIST 0: MBIST execution still ongoing 1: MBIST execution finished

82.5.24 STCU2 Off-Line MBIST End Flag High Register (STCU_MBEH)

The STCU_MBEH register includes the End Flag related to the execution of the **selected** Off-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Figure 2070. STCU2 Off-Line MBIST End Flag High Register (STCU_MBEH)

Offset: 0058h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE95	MBE94	MBE93	MBE92	MBE91	MBE90	MBE89	MBE88	MBE87	MBE86	MBE85	MBE84	MBE83	MBE82	MBE81	MBE80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE79	MBE78	MBE77	MBE76	MBE75	MBE74	MBE73	MBE72	MBE71	MBE70	MBE69	MBE68	MBE67	MBE66	MBE65	MBE64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1929. STCU_MBEH field descriptions

Field	Description
MBEx	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0: MBIST execution still ongoing 1: MBIST execution finished

82.5.25 STCU2 On-Line MBIST Status Low Register (STCU_MBSLSW)

The STCU_MBSLSW register includes the results corresponding to the execution of the **selected** On-Line MBIST in the range NMCUT = 0..31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2071. STCU2 On-Line MBIST Status Low Register (STCU_MBSLSW)

Offset: 005Ch

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW31	MBSSW30	MBSSW29	MBSSW28	MBSSW27	MBSSW26	MBSSW25	MBSSW24	MBSSW23	MBSSW22	MBSSW21	MBSSW20	MBSSW19	MBSSW18	MBSSW17	MBSSW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15	MBSSW14	MBSSW13	MBSSW12	MBSSW11	MBSSW10	MBSSW9	MBSSW8	MBSSW7	MBSSW6	MBSSW5	MBSSW4	MBSSW3	MBSSW2	MBSSW1	MBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1930. STCU_MBSLSW field descriptions

Field	Description
MBSSWx	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0: Failed NMCUT BIST execution 1: No Fault detected during the NMCUT BIST execution

82.5.26 STCU2 On-Line MBIST Status Medium Register (STCU_MBSMSW)

The STCU_MBSMSW register includes the results corresponding to the execution of the **selected** On-Line MBIST in the range NMCUT = 32..63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2072. STCU2 On-Line MBIST Status Medium Register (STCU_MBSMSW)

Offset: 0060h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW63	MBSSW62	MBSSW61	MBSSW60	MBSSW59	MBSSW58	MBSSW57	MBSSW56	MBSSW55	MBSSW54	MBSSW53	MBSSW52	MBSSW51	MBSSW50	MBSSW49	MBSSW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW47	MBSSW46	MBSSW45	MBSSW44	MBSSW43	MBSSW42	MBSSW41	MBSSW40	MBSSW39	MBSSW38	MBSSW37	MBSSW36	MBSSW35	MBSSW34	MBSSW33	MBSSW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1931. STCU_MBSMSW field descriptions

Field	Description
MBSSWx	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0: Failed NMCUT BIST execution 1: No Fault detected during the NMCUT BIST execution

82.5.27 STCU2 On-Line MBIST Status High Register (STCU_MBSHSW)

The STCU_MBSHSW register includes the results corresponding to the execution of the **selected** On-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2073. STCU2 On-Line MBIST Status High Register (STCU_MBSHSW)

Offset: 0064h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW95	MBSSW94	MBSSW93	MBSSW92	MBSSW91	MBSSW90	MBSSW89	MBSSW88	MBSSW87	MBSSW86	MBSSW85	MBSSW84	MBSSW83	MBSSW82	MBSSW81	MBSSW80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW79	MBSSW78	MBSSW77	MBSSW76	MBSSW75	MBSSW74	MBSSW73	MBSSW72	MBSSW71	MBSSW70	MBSSW69	MBSSW68	MBSSW67	MBSSW66	MBSSW65	MBSSW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1932. STCU_MBSHSW field descriptions

Field	Description
MBSSWx	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0: Failed NMCUT BIST execution 1: No Fault detected during the NMCUT BIST execution

82.5.28 STCU2 On-Line MBIST End Flag Low Register (STCU_MBELSW)

The STCU_MBELSW register includes the End Flag related to the execution of the **selected** On-Line MBIST in the range NMCUT = 0..31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2074. STCU2 On-Line MBIST End Flag Low Register (STCU_MBELSW)

Offset: 0068h Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW31	MBESW30	MBESW29	MBESW28	MBESW27	MBESW26	MBESW25	MBESW24	MBESW23	MBESW22	MBESW21	MBESW20	MBESW19	MBESW18	MBESW17	MBESW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW15	MBESW14	MBESW13	MBESW12	MBESW11	MBESW10	MBESW9	MBESW8	MBESW7	MBESW6	MBESW5	MBESW4	MBESW3	MBESW2	MBESW1	MBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1933. STCU_MBELSW field descriptions

Field	Description
MBESWx	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0: MBIST execution still ongoing 1: MBIST execution finished

82.5.29 STCU2 On-Line MBIST End Flag Medium Register (STCU_MBEMSW)

The STCU_MBEMSW register includes the End Flag related to the execution of the **selected** On-Line MBIST in the range NMCUT = 32...63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2075. STCU2 On-Line MBIST End Flag Medium Register (STCU_MBEMSW)

Offset: 006Ch

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW63	MBESW62	MBESW61	MBESW60	MBESW59	MBESW58	MBESW57	MBESW56	MBESW55	MBESW54	MBESW53	MBESW52	MBESW51	MBESW50	MBESW49	MBESW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW47	MBESW46	MBESW45	MBESW44	MBESW43	MBESW42	MBESW41	MBESW40	MBESW39	MBESW38	MBESW37	MBESW36	MBESW35	MBESW34	MBESW33	MBESW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1934. STCU_MBEMSW field descriptions

Field	Description
MBESWx	<i>On-Line End status (32 to 63) of the selected MBIST</i> 0: MBIST execution still ongoing 1: MBIST execution finished

82.5.30 STCU2 On-Line MBIST End Flag High Register (STCU_MBEHSW)

The STCU_MBEHSW register includes the End Flag related to the execution of the **selected** On-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2076. STCU2 On-Line MBIST End Flag High Register (STCU_MBEHSW)

Offset: 0070h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW95	MBESW94	MBESW93	MBESW92	MBESW91	MBESW90	MBESW89	MBESW88	MBESW87	MBESW86	MBESW85	MBESW84	MBESW83	MBESW82	MBESW81	MBESW80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW79	MBESW78	MBESW77	MBESW76	MBESW75	MBESW74	MBESW73	MBESW72	MBESW71	MBESW70	MBESW69	MBESW68	MBESW67	MBESW66	MBESW65	MBESW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1935. STCU_MBEHSW field descriptions

Field	Description
MBESWx	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0: MBIST execution still ongoing 1: MBIST execution finished

82.5.31 STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)

The STCU_MBUFML register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 0..31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2077. STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)

Offset: 0074h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM31	MBUFM30	MBUFM29	MBUFM28	MBUFM27	MBUFM26	MBUFM25	MBUFM24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Table 1936. STCU_MBUFML field descriptions

Field	Description
MBUFMx	<i>MBIST Unrecoverable Fault Mapping</i> 0: Recoverable Fault mapping 1: Unrecoverable Fault mapping

82.5.32 STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)

The STCU_MBUFMM register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 32..63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2078. STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)

Offset: 0078h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM63	MBUFM62	MBUFM61	MBUFM60	MBUFM59	MBUFM58	MBUFM57	MBUFM56	MBUFM55	MBUFM54	MBUFM53	MBUFM52	MBUFM51	MBUFM50	MBUFM49	MBUFM48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM47	MBUFM46	MBUFM45	MBUFM44	MBUFM43	MBUFM42	MBUFM41	MBUFM40	MBUFM39	MBUFM38	MBUFM37	MBUFM36	MBUFM35	MBUFM34	MBUFM33	MBUFM32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Table 1937. STCU_MBUFMM field descriptions

Field	Description
MBUFMx	<i>MBIST Unrecoverable Fault Mapping</i> 0: Recoverable Fault mapping 1: Unrecoverable Fault mapping

82.5.33 STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)

The STCU_MBUFMH register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2079. STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)

Offset: 007Ch

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM95	MBUFM94	MBUFM93	MBUFM92	MBUFM91	MBUFM90	MBUFM89	MBUFM88	MBUFM87	MBUFM86	MBUFM85	MBUFM84	MBUFM83	MBUFM82	MBUFM81	MBUFM80
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM79	MBUFM78	MBUFM77	MBUFM76	MBUFM75	MBUFM74	MBUFM73	MBUFM72	MBUFM71	MBUFM70	MBUFM69	MBUFM68	MBUFM67	MBUFM66	MBUFM65	MBUFM64
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1938. STCU_MBUFMH field descriptions

Field	Description
MBUFMx	<i>MBIST Unrecoverable Fault Mapping</i> 0: Recoverable Fault mapping 1: Unrecoverable Fault mapping

82.5.34 STCU2 LBIST Control Register (STCU_LB_CTRL)

The STCU_LB_CTRL register defines the control setting of each LBIST controller.

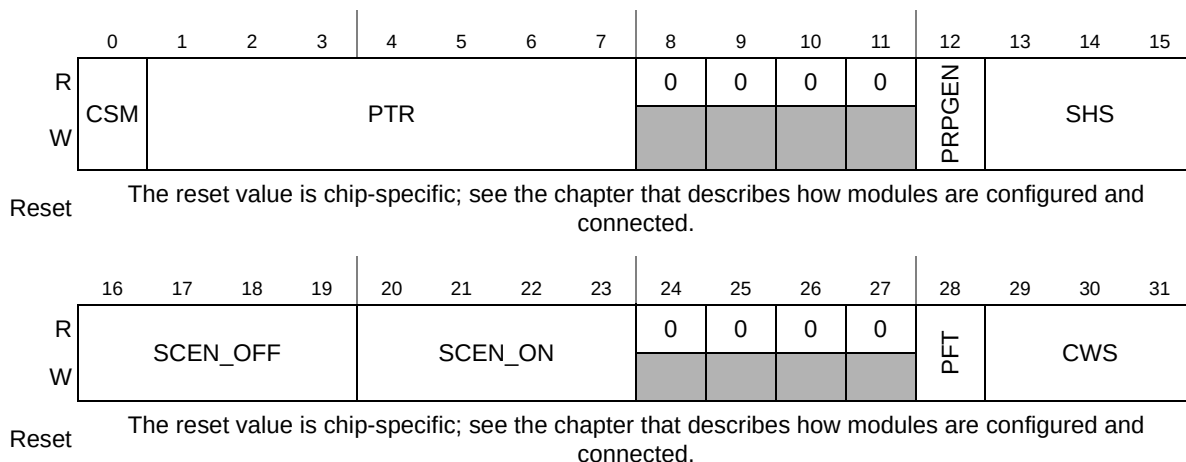
The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2080. STCU2 LBIST Control Register (STCU_LB_CTRL)

Offset: 0100h + (NLBIST – 1) × 40h

Access: User read/write

**Table 1939. STCU_LB_CTRL field descriptions**

Field	Description
CSM	<i>Concurrent/sequential mode</i> 0 Sequential mode 1 Concurrent mode
PTR	<i>Next LBIST or MBIST pointer</i> PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to 1, otherwise it's scheduled sequentially to the completion of the current LBIST execution. In case of NIL pointer the CSM bit has to be set Sequential (0) to define this is the end of the list. The self testing procedure is stopped when the current LBIST has been completed. 0h to (NLBIST -1): pointer to LBIST 10h to (10h + MBIST -1b): pointer to MBIST 7Fh: pointer to NIL. No BIST execution. others: invalid pointer => an error is set into the STCU_ERR register.
PRPGEN	<i>Enable PRPG Loading</i> 0: Default LBIST value of the PRPG is used during LBIST run 1: LBIST PRPG value is initialized with the STCU_LB_PRPGL/H
SHS	<i>Shift speed</i> SHS defines the shift speed 000: Shift at full rate (bist_clk) 001: Shift at 1/2 rate (bist_clk) 010: Shift at 1/3 rate (bist_clk) 011: Shift at 1/4 rate (bist_clk) 100: Shift at 1/5 rate (bist_clk) 101: Shift at 1/6rate (bist_clk) 110: Shift at 1/7rate (bist_clk) 111: Shift at 1/8 rate (bist_clk)

Table 1939. STCU_LB_CTRL field descriptions (continued)

Field	Description
SCEN_OFF	<i>Scan enable OFF</i> SCEN_OFF defines the number of clock cycles OFF following the falling transition on the SCEN 0000: 0 delay cycles 0001: 1 delay cycles ... 1111: 15 delay cycles
SCEN_ON	<i>Scan enable ON</i> SCEN_ON defines the number of clock cycles OFF following the rising transition on the SCEN 0000: 0 delay cycles 0001: 1 delay cycles ... 1111: 15 delay cycles
PFT	<i>Past Flush Test</i> The LBIST controller, by default apply 32 Flush Test patterns. This bit allows to skip/apply the Flush test pattern sequence. 0: Apply Flush Test Patterns 1: Skip Flush Test Patterns
CWS	<i>Capture window size</i> CWS defines the capture window size. 000: illegal 001: controller waits 1 shift cycle for capture to finish 010: controller waits 2 shift cycles for capture to finish ... 111: controller waits 7 shift cycles for capture to finish

82.5.35 STCU2 LBIST PC Stop Register (STCU_LB_PCS)

The STCU_LB_PCS register defines the pattern counter stop of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2081. STCU2 LBIST PC Stop Register (STCU_LB_PCS)

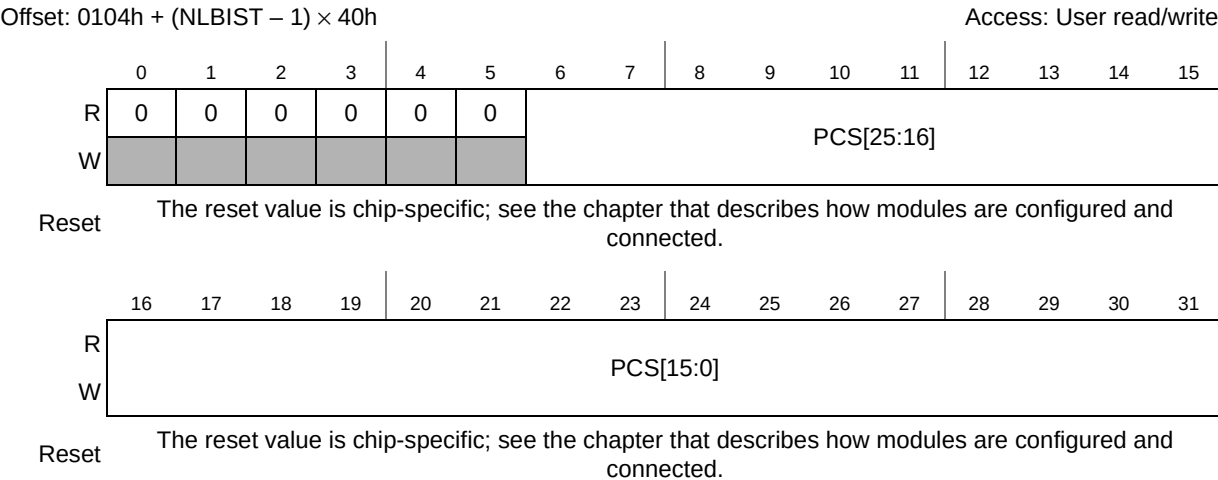


Table 1940. STCU_LB_PCS field descriptions

Field	Description
PCS	<i>Pattern counter stop</i> PCS defines the pattern counter stop value.

82.5.36 STCU2 LBIST PRPG Low Register (STCU_LB_PRPGL)

The STCU_LB_PRPGL register defines the LSB part (31..0) of the PRPG start value of the LBIST controller.

The size of the register depends on the number of PRPG bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2082. STCU2 LBIST PRPG Low Register (STCU_LB_PRPGL)

Offset: 0108h + (NLBIST – 1) × 40h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRPG31	PRPG30	PRPG29	PRPG28	PRPG27	PRPG26	PRPG25	PRPG24	PRPG23	PRPG22	PRPG21	PRPG20	PRPG19	PRPG18	PRPG17	PRPG16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRPG15	PRPG14	PRPG13	PRPG12	PRPG11	PRPG10	PRPG9	PRPG8	PRPG7	PRPG6	PRPG5	PRPG4	PRPG3	PRPG2	PRPG1	PRPG0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Table 1941. STCU_LB_PRPGL field descriptions

Field	Description
PRPGx	<i>PRPG LBIST Low Bits</i> This field defines the Low part (31..0) of the LBIST PRPG start value.

82.5.37 STCU2 LBIST PRPG High Register (STCU_LB_PRPGH)

The STCU_LB_PRPGH register defines the MSB part (63..32) of the PRPG start value of the LBIST controller.

The size of the register depends on the number of PRPG bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2083. STCU2 LBIST PRPG High Register (STCU_LB_PRPGH)

Offset: 010Ch + (NLBIST – 1) × 40h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRPG63	PRPG62	PRPG61	PRPG60	PRPG59	PRPG58	PRPG57	PRPG56	PRPG55	PRPG54	PRPG53	PRPG52	PRPG51	PRPG50	PRPG49	PRPG48
W																

Reset

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRPG47	PRPG46	PRPG45	PRPG44	PRPG43	PRPG42	PRPG41	PRPG40	PRPG39	PRPG38	PRPG37	PRPG36	PRPG35	PRPG34	PRPG33	PRPG32
W																

Reset

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1942. STCU_LB_PRPGH field descriptions

Field	Description
PRPGx	<i>PRPG LBIST High Bits</i> This field is the High part (63..32) of the LBIST PRPG start value.

82.5.38 STCU2 Off-Line LBIST MISR Expected Low Register (STCU_LB_MISREL)

The STCU_LB_MISREL register defines the LSB part (31..0) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Figure 2084. STCU2 Off-Line LBIST MISR Expected Low Register (STCU_LB_MISREL)

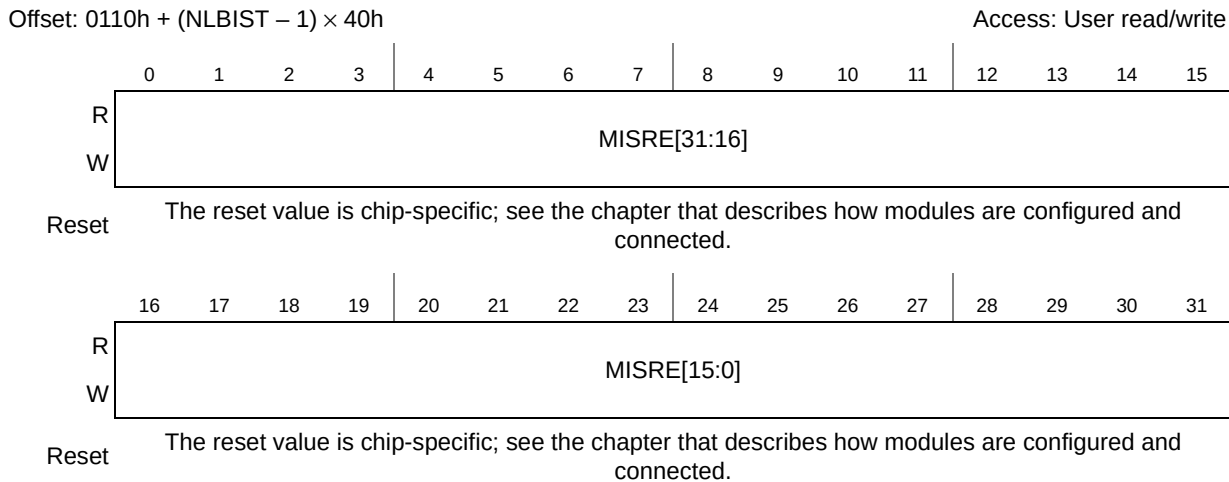


Table 1943. STCU_LB_MISREL field descriptions

Field	Description
MISREx	<i>Off-Line MISR Expected low Bits</i> This field defines the low part (31..0) of the Expected MISR.

82.5.39 STCU2 Off-Line LBIST MISR Expected High Register (STCU_LB_MISREH)

The STCU_LB_MISREH register defines the MSB part (63..32) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Figure 2085. STCU2 LBIST MISR Expected High Register (STCU_LB_MISREH)

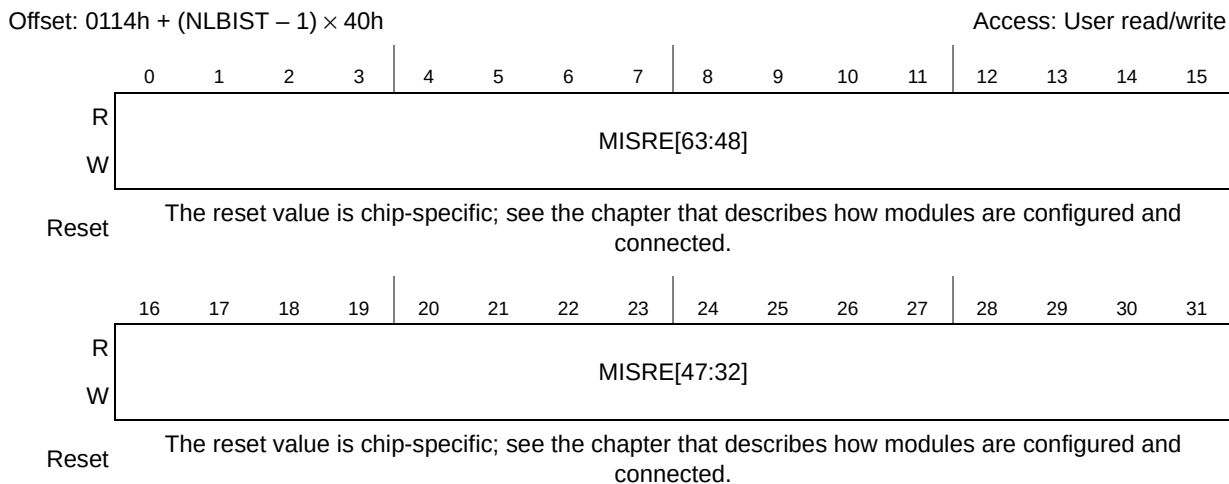


Table 1944. STCU_LB_MISREH field descriptions

Field	Description
MISREx	<i>Off-Line MISR Expected high Bits</i> This field defines the high part (63..32) of the Expected MISR.

82.5.40 STCU2 Off-Line LBIST MISR Read Low Register (STCU_LB_MISRRL)

The STCU_LB_MISRRL register report the LSB part (31..0) of the MISR obtained at the end of the Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Figure 2086. STCU2 Off-Line LBIST MISR Read Low Register (STCU_LB_MISRRL)

Offset: 0118h + (NLBIST – 1) × 40h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRR31	MISRR30	MISRR29	MISRR28	MISRR27	MISRR26	MISRR25	MISRR24	MISRR23	MISRR22	MISRR21	MISRR20	MISRR19	MISRR18	MISRR17	MISRR16
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRR15	MISRR14	MISRR13	MISRR12	MISRR11	MISRR10	MISRR9	MISRR8	MISRR7	MISRR6	MISRR5	MISRR4	MISRR3	MISRR2	MISRR1	MISRR0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1945. STCU_LB_MISRRL field descriptions

Field	Description
MISRRLx	<i>Off-Line MISR Read Low Bits</i> This field is equivalent to the Low Bits (31..0) of the MISR obtained at the end of the Off-Line LBIST execution

82.5.41 STCU2 Off-Line LBIST MISR Read High Register (STCU_LB_MISRRLH)

The STCU_LB_MISRRLH register report the MSB part (63..32) of the MISR obtained at the end of each Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Figure 2087. STCU2 Off-Line LBIST MISR Read High Register (STCU_LB_MISRRH)

Offset: 011Ch + (NLBIST – 1) × 40h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRR63	MISRR62	MISRR61	MISRR60	MISRR59	MISRR58	MISRR57	MISRR56	MISRR55	MISRR54	MISRR53	MISRR52	MISRR51	MISRR50	MISRR49	MISRR48
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRR47	MISRR46	MISRR45	MISRR44	MISRR43	MISRR42	MISRR41	MISRR40	MISRR39	MISRR38	MISRR37	MISRR36	MISRR35	MISRR34	MISRR33	MISRR32
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1946. STCU_LB_MISRRH field descriptions

Field	Description
MISRRx	Off-Line MISR Read High Bits This field is equivalent to the High Bits (63..32) of the MISR obtained at the end of the Off-Line LBIST execution

82.5.42 STCU2 On-Line LBIST MISR Expected Low Register (STCU_LB_MISRELSW)

The STCU_LB_MISRELSW register defines the LSB part (31..0) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2088. STCU2 On-Line LBIST MISR Expected Low Register (STCU_LB_MISRELSW)

Offset: 0120h + (NLBIST – 1) × 40h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRESW31	MISRESW30	MISRESW29	MISRESW28	MISRESW27	MISRESW26	MISRESW25	MISRESW24	MISRESW23	MISRESW22	MISRESW21	MISRESW20	MISRESW19	MISRESW18	MISRESW17	MISRESW16
W	MISRESW31	MISRESW30	MISRESW29	MISRESW28	MISRESW27	MISRESW26	MISRESW25	MISRESW24	MISRESW23	MISRESW22	MISRESW21	MISRESW20	MISRESW19	MISRESW18	MISRESW17	MISRESW16

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRESW15	MISRESW14	MISRESW13	MISRESW12	MISRESW11	MISRESW10	MISRESW9	MISRESW8	MISRESW7	MISRESW6	MISRESW5	MISRESW4	MISRESW3	MISRESW2	MISRESW1	MISRESW0
W	MISRESW15	MISRESW14	MISRESW13	MISRESW12	MISRESW11	MISRESW10	MISRESW9	MISRESW8	MISRESW7	MISRESW6	MISRESW5	MISRESW4	MISRESW3	MISRESW2	MISRESW1	MISRESW0

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1947. STCU_LB_MISRELSW field descriptions

Field	Description
MISRESWx	<i>On-Line MISR Expected low Bits</i> This field defines the low part (31..0) of the Expected MISR.

82.5.43 STCU2 On-Line LBIST MISR Expected High Register (STCU_LB_MISREHSW)

The STCU_LB_MISREHSW register defines the MSB part (63..32) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2089. STCU2 On-Line LBIST MISR Expected High Register (STCU_LB_MISREHSW)

Offset: 0124h + (NLBIST – 1) × 40h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRESW63	MISRESW62	MISRESW61	MISRESW60	MISRESW59	MISRESW58	MISRESW57	MISRESW56	MISRESW55	MISRESW54	MISRESW53	MISRESW52	MISRESW51	MISRESW50	MISRESW49	MISRESW48
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRESW47	MISRESW46	MISRESW45	MISRESW44	MISRESW43	MISRESW42	MISRESW41	MISRESW40	MISRESW39	MISRESW38	MISRESW37	MISRESW36	MISRESW35	MISRESW34	MISRESW33	MISRESW32
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1948. STCU_LB_MISREHSW field descriptions

Field	Description
MISRESWx	<i>On-Line MISR Expected high Bits</i> This field defines the high part (63..32) of the Expected MISR.

82.5.44 STCU2 On-Line LBIST MISR Read Low Register (STCU_LB_MISRRLSW)

The STCU_LB_MISRRLSW register report the LSB part (31..0) of the MISR obtained at the end of the On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2090. STCU2 On-Line LBIST MISR Read Low Register (STCU_LB_MISRRLSW)

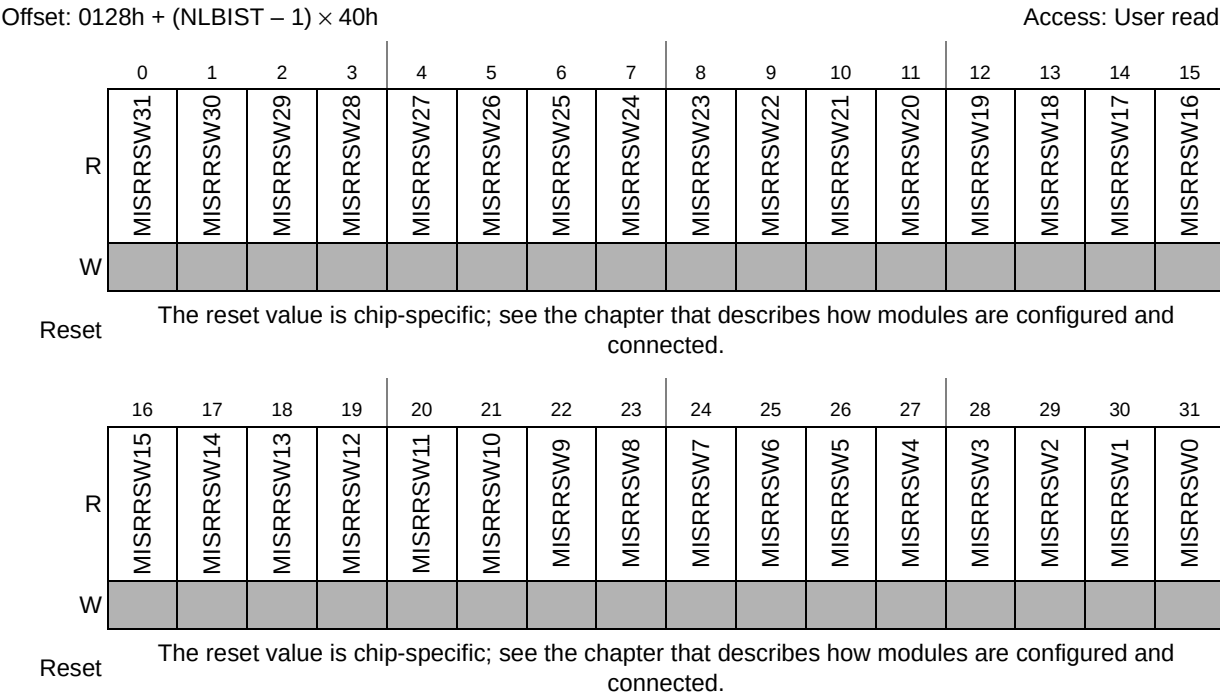


Table 1949. STCU_LB_MISRRLSW field descriptions

Field	Description
MISRRSWx	On-Line MISR Read Low Bin This field is equivalent to the Low Bits (31..0) of the MISR obtained at the end of the Off-Line LBIST execution.

82.5.45 STCU2 On-Line LBIST MISR Read High Register (STCU_LB_MISRRHSW)

The STCU_LB_MISRRHSW register report the MSB part (63..32) of the MISR obtained at the end of each On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2091. STCU2 On-Line LBIST MISR Read High Register (STCU_LB_MISRRHSW)

Offset: 012Ch + (NLBIST – 1) × 40h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRRSW63	MISRRSW62	MISRRSW61	MISRRSW60	MISRRSW59	MISRRSW58	MISRRSW57	MISRRSW56	MISRRSW55	MISRRSW54	MISRRSW53	MISRRSW52	MISRRSW51	MISRRSW50	MISRRSW49	MISRRSW48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSW47	MISRRSW46	MISRRSW45	MISRRSW44	MISRRSW43	MISRRSW42	MISRRSW41	MISRRSW40	MISRRSW39	MISRRSW38	MISRRSW37	MISRRSW36	MISRRSW35	MISRRSW34	MISRRSW33	MISRRSW32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Table 1950. STCU_LB_MISRRHSW field descriptions

Field	Description
MISRRSWx	On-Line MISR Read High Bits This field is equivalent to the Low Bits (63..32) of the MISR obtained at the end of the Off-Line LBIST execution.

82.5.46 STCU2 MBIST Control Register (STCU_MB_CTRL)

The STCU_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0

Figure 2092. STCU2 MBIST Control Register (STCU_MB_CTRL)

Offset: 0600h + (NMCUT – 1) × 4h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSM	PTR							0	0	0	0	0	0	0	0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																

Reset The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Table 1951. STCU_MB_CTRL field descriptions

Field	Description
CSM	<i>Concurrent/sequential mode</i> 0: Sequential mode 1: Concurrent mode
PTR	<i>Next LBIST or MBIST pointer</i> PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to 1, otherwise it's scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer the CSM bit has to be set Sequential (0) to define this is the end of the list. The self testing procedure is stopped when the current MBIST has been completed. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1b): pointer to MBIST 7Fh: pointer to NIL. No BIST execution. others: invalid pointer => an error is set into the STCU_ERR register.

82.6 Register map

In the following is reported the STCU2 register map in case a full implementation is performed. In case a small number of LBIST or MBIST are implemented the following considerations have to be taken into account:

- the size of the STCU_LBS, STCU_LBSSW, STCU_LBE, STCU_LBESW, STCU_LBRMSW, STCU_LBUFM depends on the number of LBIST;
- the size of STCU_MBSL, STCU_MBSLSW, STCU_MBEL, STCU_MBELSW, STCU_MBUFML depends on the number of BISTed RAMs/ROMs;
- the size of STCU_MBSM, STCU_MBSMSW, STCU_MBEM, STCU_MBEMSW, STCU_MBUFMM depends on the number of BISTed RAMs/ROMs only when the related value is greater than 32, otherwise these registers are not implemented;
- the size of the STCU_MBSH, STCU_MBSHSW, STCU_MBEH, STCU_MBEHSW, STCU_MBUFMH depends on the number of BISTed RAMs/ROMs only when the related value is greater than 64, otherwise these registers are not implemented;
- the number of STCU_LB_CTRL, STCU_LB_PCS, STCU_LB_PRPGL, STCU_LB_PRPGH, STCU_LB_MISREL, STCU_LB_MISRELSW, STCU_LB_MISREH, STCU_LB_MISREHSW, STCU_LB_MISRRL, STCU_LB_MISRRLSW, STCU_LB_MISRRLH and STCU_LB_MISRRLHSW depends on the number of LBIST;
- the size of each STCU_LB_MISREL, STCU_LB_MISRELSW and STCU_LB_MISRRL, STCU_LB_MISRRLSW depends on the number of MISR bits of the related LBIST;
- the size of each STCU_LB_MISREH, STCU_LB_MISREHSW, and STCU_LB_MISRRLH, STCU_LB_MISRRLHSW, depends on the number of MISR bits of the related LBIST only when the related value is greater than 32 otherwise the registers are not implemented;
- the size of STCU_LB_PRPGL and STCU_LB_PRPGH depends on the number of PRPG bits of the related LBIST.
- the number of STCU_MB_CTRL depends on the number of BISTed RAMs/ROMs.

Table 1952. STCU2 register map

Addr. Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
0000h	STCU_RUN	Reserved																				B Y P	M B P L L E N	L B P L L E N	Reserved								R U N		
0004h	STCU_RUNSW	Reserved																				M B I E	L B I E	M B S W P L L E N	L B S W P L L E N	Reserved								R U N S W _ A B O R T	R U N S W
0008h	STCU_SKC	SKC																																	

Table 1952. STCU2 register map (continued)

Addr. Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
000Ch	STCU_CFG	R e s e r v e d	PTR							LB_DELAY							Reserved							W R P	R e s e r v e d	C R C E N	P M O S E N	M B U	CLK_CFG				
0010h	STCU_PLL_CFG	R e s e r v e d	P L L O D F 5	P L L O D F 4	P L L O D F 3	P L L O D F 2	P L L O D F 1	P L L O D F 0	Reserved					P L L I D F 2	P L L I D F 1	P L L I D F 0	Reserved							P L L L D F 6	P L L L D F 5	P L L L D F 4	P L L L D F 3	P L L L D V 2	P L L L D V 1	P L L L D V 0			
0014h	STCU_WDG	WDGEOC																															
0018h	STCU_INT_FLG	Reserved																													M B I F L G	L B I F L G	
001Ch	STCU_CRCE	CRCE																															
0020h	STCU_CRCR	CRCR																															
0024h	STCU_ERR_STAT	Reserved						A B O R T S W	A B O R T S W	Reserved				L O C K E S W	W D T O S W	C R C S S W	E N G E S W	I N V P S W	Reserved					U F S F	R F S F	Reserved			L O C K E	W D T O	C R C S	E N G E	I N V P
0028h	STCU_ERR_FM	Reserved																									L O C K E U F M	W D T O U F M	C R C S U F M	E N G E U F M	I N V P U F M		
002Ch	STCU_LBS	Reserved															L B S 1 5	L B S 1 4	L B S 1 3	L B S 1 2	L B S 1 1	L B S 1 0	L B S 9	L B S 8	L B S 7	L B S 6	L B S 5	L B S 4	L B S 3	L B S 2	L B S 1	L B S 0	
0030h	STCU_LBE	Reserved															L B E 1 5	L B E 1 4	L B E 1 3	L B E 1 2	L B E 1 1	L B E 1 0	L B E 9	L B E 8	L B E 7	L B E 6	L B E 5	L B E 4	L B E 3	L B E 2	L B E 1	L B E 0	
0034h	STCU_LBSSW	Reserved															L B S S W 1 5	L B S S W 1 4	L B S S W 1 3	L B S S W 1 2	L B S S W 1 1	L B S S W 1 0	L B S S W 9	L B S S W 8	L B S S W 7	L B S S W 6	L B S S W 5	L B S S W 4	L B S S W 3	L B S S W 2	L B S S W 1	L B S S W 0	
0038h	STCU_LBESW	Reserved															L B E S W 1 5	L B E S W 1 4	L B E S W 1 3	L B E S W 1 2	L B E S W 1 1	L B E S W 1 0	L B E S W 9	L B E S W 8	L B E S W 7	L B E S W 6	L B E S W 5	L B E S W 4	L B E S W 3	L B E S W 2	L B E S W 1	L B E S W 0	

Table 1952. STCU2 register map (continued)

Addr. Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
003Ch	STCU_LBRMSW	Reserved																	L B R M S W 1 5	L B R M S W 1 4	L B R M S W 1 3	L B R M S W 1 2	L B R M S W 1 1	L B R M S W 1 0	L B R M S W 9	L B R M S W 8	L B R M S W 7	L B R M S W 6	L B R M S W 5	L B R M S W 4	L B R M S W 3	L B R M S W 2	L B R M S W 1	L B R M S W 0
0040h	STCU_LBUFM	Reserved																	L B U F M 1 5	L B U F M 1 4	L B U F M 1 3	L B U F M 1 2	L B U F M 1 1	L B U F M 1 0	L B U F M 9	L B U F M 8	L B U F M 7	L B U F M 6	L B U F M 5	L B U F M 4	L B U F M 3	L B U F M 2	L B U F M 1	L B U F M 0
0044h	STCU_MBSL	M B S 3 1	M B S 3 0	M B S 2 9	M B S 2 8	M B S 2 7	M B S 2 6	M B S 2 5	M B S 2 4	M B S 2 3	M B S 2 2	M B S 2 1	M B S 2 0	M B S 1 9	M B S 1 8	M B S 1 7	M B S 1 6	M B S 1 5	M B S 1 4	M B S 1 3	M B S 1 2	M B S 1 1	M B S 1 0	M B S 9	M B S 8	M B S 7	M B S 6	M B S 5	M B S 4	M B S 3	M B S 2	M B S 1	M B S 0	
0048h	STCU_MBSM	M B S 6 3	M B S 6 2	M B S 6 1	M B S 6 0	M B S 5 9	M B S 5 8	M B S 5 7	M B S 5 6	M B S 5 5	M B S 5 4	M B S 5 3	M B S 5 2	M B S 5 1	M B S 5 0	M B S 4 9	M B S 4 8	M B S 4 7	M B S 4 6	M B S 4 5	M B S 4 4	M B S 4 3	M B S 4 2	M B S 4 1	M B S 4 0	M B S 3 9	M B S 3 8	M B S 3 7	M B S 3 6	M B S 3 5	M B S 3 4	M B S 3 3	M B S 3 2	
004Ch	STCU_MBSH	M B S 9 5	M B S 9 4	M B S 9 3	M B S 9 2	M B S 9 1	M B S 9 0	M B S 8 9	M B S 8 8	M B S 8 7	M B S 8 6	M B S 8 5	M B S 8 4	M B S 8 3	M B S 8 2	M B S 8 1	M B S 8 0	M B S 7 9	M B S 7 8	M B S 7 7	M B S 7 6	M B S 7 5	M B S 7 4	M B S 7 3	M B S 7 2	M B S 7 1	M B S 7 0	M B S 6 9	M B S 6 8	M B S 6 7	M B S 6 6	M B S 6 5	M B S 6 4	
0050h	STCU_MBEL	M B E 3 1	M B E 3 0	M B E 2 9	M B E 2 8	M B E 2 7	M B E 2 6	M B E 2 5	M B E 2 4	M B E 2 3	M B E 2 2	M B E 2 1	M B E 2 0	M B E 1 9	M B E 1 8	M B E 1 7	M B E 1 6	M B E 1 5	M B E 1 4	M B E 1 3	M B E 1 2	M B E 1 1	M B E 1 0	M B E 9	M B E 8	M B E 7	M B E 6	M B E 5	M B E 4	M B E 3	M B E 2	M B E 1	M B E 0	
0054h	STCU_MBEM	M B E 6 3	M B E 6 2	M B E 6 1	M B E 6 0	M B E 5 9	M B E 5 8	M B E 5 7	M B E 5 6	M B E 5 5	M B E 5 4	M B E 5 3	M B E 5 2	M B E 5 1	M B E 5 0	M B E 4 9	M B E 4 8	M B E 4 7	M B E 4 6	M B E 4 5	M B E 4 4	M B E 4 3	M B E 4 2	M B E 4 1	M B E 4 0	M B E 3 9	M B E 3 8	M B E 3 7	M B E 3 6	M B E 3 5	M B E 3 4	M B E 3 3	M B E 3 2	
0058h	STCU_MBEH	M B E 9 5	M B E 9 4	M B E 9 3	M B E 9 2	M B E 9 1	M B E 9 0	M B E 8 9	M B E 8 8	M B E 8 7	M B E 8 6	M B E 8 5	M B E 8 4	M B E 8 3	M B E 8 2	M B E 8 1	M B E 8 0	M B E 7 9	M B E 7 8	M B E 7 7	M B E 7 6	M B E 7 5	M B E 7 4	M B E 7 3	M B E 7 2	M B E 7 1	M B E 7 0	M B E 6 9	M B E 6 8	M B E 6 7	M B E 6 6	M B E 6 5	M B E 6 4	
005Ch	STCU_MBSLSW	M B S S W 3 1	M B S S W 3 0	M B S S W 2 9	M B S S W 2 8	M B S S W 2 7	M B S S W 2 6	M B S S W 2 5	M B S S W 2 4	M B S S W 2 3	M B S S W 2 2	M B S S W 2 1	M B S S W 2 0	M B S S W 1 9	M B S S W 1 8	M B S S W 1 7	M B S S W 1 6	M B S S W 1 5	M B S S W 1 4	M B S S W 1 3	M B S S W 1 2	M B S S W 1 1	M B S S W 1 0	M B S S W 9	M B S S W 8	M B S S W 7	M B S S W 6	M B S S W 5	M B S S W 4	M B S S W 3	M B S S W 2	M B S S W 1	M B S S W 0	
0060h	STCU_MBSMSW	M B S S W 6 3	M B S S W 6 2	M B S S W 6 1	M B S S W 6 0	M B S S W 5 9	M B S S W 5 8	M B S S W 5 7	M B S S W 5 6	M B S S W 5 5	M B S S W 5 4	M B S S W 5 3	M B S S W 5 2	M B S S W 5 1	M B S S W 5 0	M B S S W 4 9	M B S S W 4 8	M B S S W 4 7	M B S S W 4 6	M B S S W 4 5	M B S S W 4 4	M B S S W 4 3	M B S S W 4 2	M B S S W 4 1	M B S S W 4 0	M B S S W 3 9	M B S S W 3 8	M B S S W 3 7	M B S S W 3 6	M B S S W 3 5	M B S S W 3 4	M B S S W 3 3	M B S S W 3 2	
0064h	STCU_MBSHSW	M B S S W 9 5	M B S S W 9 4	M B S S W 9 3	M B S S W 9 2	M B S S W 9 1	M B S S W 9 0	M B S S W 8 9	M B S S W 8 8	M B S S W 8 7	M B S S W 8 6	M B S S W 8 5	M B S S W 8 4	M B S S W 8 3	M B S S W 8 2	M B S S W 8 1	M B S S W 8 0	M B S S W 7 9	M B S S W 7 8	M B S S W 7 7	M B S S W 7 6	M B S S W 7 5	M B S S W 7 4	M B S S W 7 3	M B S S W 7 2	M B S S W 7 1	M B S S W 7 0	M B S S W 6 9	M B S S W 6 8	M B S S W 6 7	M B S S W 6 6	M B S S W 6 5	M B S S W 6 4	

Table 1952. STCU2 register map (continued)

Addr. Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
0068h	STCU_MBELSW	M B E S W 3 1	M B E S W 3 0	M B E S W 2 9	M B E S W 2 8	M B E S W 2 7	M B E S W 2 6	M B E S W 2 5	M B E S W 2 4	M B E S W 2 3	M B E S W 2 2	M B E S W 2 1	M B E S W 2 0	M B E S W 1 9	M B E S W 1 8	M B E S W 1 7	M B E S W 1 6	M B E S W 1 5	M B E S W 1 4	M B E S W 1 3	M B E S W 1 2	M B E S W 1 1	M B E S W 1 0	M B E S W 9	M B E S W 8	M B E S W 7	M B E S W 6	M B E S W 5	M B E S W 4	M B E S W 3	M B E S W 2	M B E S W 1	M B E S W 0			
006Ch	STCU_MBEMSW	M B E S W 6 3	M B E S W 6 2	M B E S W 6 1	M B E S W 6 0	M B E S W 5 9	M B E S W 5 8	M B E S W 5 7	M B E S W 5 6	M B E S W 5 5	M B E S W 5 4	M B E S W 5 3	M B E S W 5 2	M B E S W 5 1	M B E S W 5 0	M B E S W 4 9	M B E S W 4 8	M B E S W 4 7	M B E S W 4 6	M B E S W 4 5	M B E S W 4 4	M B E S W 4 3	M B E S W 4 2	M B E S W 4 1	M B E S W 4 0	M B E S W 3 9	M B E S W 3 8	M B E S W 3 7	M B E S W 3 6	M B E S W 3 5	M B E S W 3 4	M B E S W 3 3	M B E S W 3 2			
0070h	STCU_MBEHSW	M B E S W 9 5	M B E S W 9 4	M B E S W 9 3	M B E S W 9 2	M B E S W 9 1	M B E S W 9 0	M B E S W 8 9	M B E S W 8 8	M B E S W 8 7	M B E S W 8 6	M B E S W 8 5	M B E S W 8 4	M B E S W 8 3	M B E S W 8 2	M B E S W 8 1	M B E S W 8 0	M B E S W 7 9	M B E S W 7 8	M B E S W 7 7	M B E S W 7 6	M B E S W 7 5	M B E S W 7 4	M B E S W 7 3	M B E S W 7 2	M B E S W 7 1	M B E S W 7 0	M B E S W 6 9	M B E S W 6 8	M B E S W 6 7	M B E S W 6 6	M B E S W 6 5	M B E S W 6 4			
0074h	STCU_MBUFML	M B U F M 3 1	M B U F M 3 0	M B U F M 2 9	M B U F M 2 8	M B U F M 2 7	M B U F M 2 6	M B U F M 2 5	M B U F M 2 4	M B U F M 2 3	M B U F M 2 2	M B U F M 2 1	M B U F M 2 0	M B U F M 1 9	M B U F M 1 8	M B U F M 1 7	M B U F M 1 6	M B U F M 1 5	M B U F M 1 4	M B U F M 1 3	M B U F M 1 2	M B U F M 1 1	M B U F M 1 0	M B U F M 9	M B U F M 8	M B U F M 7	M B U F M 6	M B U F M 5	M B U F M 4	M B U F M 3	M B U F M 2	M B U F M 1	M B U F M 0			
0078h	STCU_MBUFMM	M B U F M 6 3	M B U F M 6 2	M B U F M 6 1	M B U F M 6 0	M B U F M 5 9	M B U F M 5 8	M B U F M 5 7	M B U F M 5 6	M B U F M 5 5	M B U F M 5 4	M B U F M 5 3	M B U F M 5 2	M B U F M 5 1	M B U F M 5 0	M B U F M 4 9	M B U F M 4 8	M B U F M 4 7	M B U F M 4 6	M B U F M 4 5	M B U F M 4 4	M B U F M 4 3	M B U F M 4 2	M B U F M 4 1	M B U F M 4 0	M B U F M 3 9	M B U F M 3 8	M B U F M 3 7	M B U F M 3 6	M B U F M 3 5	M B U F M 3 4	M B U F M 3 3	M B U F M 3 2			
007Ch	STCU_MBUFMH	M B U F M 9 5	M B U F M 9 4	M B U F M 9 3	M B U F M 9 2	M B U F M 9 1	M B U F M 9 0	M B U F M 8 9	M B U F M 8 8	M B U F M 8 7	M B U F M 8 6	M B U F M 8 5	M B U F M 8 4	M B U F M 8 3	M B U F M 8 2	M B U F M 8 1	M B U F M 8 0	M B U F M 7 9	M B U F M 7 8	M B U F M 7 7	M B U F M 7 6	M B U F M 7 5	M B U F M 7 4	M B U F M 7 3	M B U F M 7 2	M B U F M 7 1	M B U F M 7 0	M B U F M 6 9	M B U F M 6 8	M B U F M 6 7	M B U F M 6 6	M B U F M 6 5	M B U F M 6 4			
Address space not used (Available for future expansion)																																				
0100h + (NLBIST -1) * 40h	STCU_LB_CTRL	C S M	PTR								Reserved				P R P G E N	SHS				SCEN_OFF				SCEN_ON				Reserved				P F T	CWS			
0104h + (NLBIST -1) * 40h	STCU_LB_PCS	Reserved								PCS																										
0108h + (NLBIST -1) * 40h	STCU_LB_PRPGL	P R P G 3 1	P R P G 3 0	P R P G 2 9	P R P G 2 8	P R P G 2 7	P R P G 2 6	P R P G 2 5	P R P G 2 4	P R P G 2 3	P R P G 2 2	P R P G 2 1	P R P G 2 0	P R P G 1 9	P R P G 1 8	P R P G 1 7	P R P G 1 6	P R P G 1 5	P R P G 1 4	P R P G 1 3	P R P G 1 2	P R P G 1 1	P R P G 1 0	P R P G 9	P R P G 8	P R P G 7	P R P G 6	P R P G 5	P R P G 4	P R P G 3	P R P G 2	P R P G 1	P R P G 0			
010Ch + (NLBIST -1) * 40h	STCU_LB_PRPGH	P R P G 6 3	P R P G 6 2	P R P G 6 1	P R P G 6 0	P R P G 5 9	P R P G 5 8	P R P G 5 7	P R P G 5 6	P R P G 5 5	P R P G 5 4	P R P G 5 3	P R P G 5 2	P R P G 5 1	P R P G 5 0	P R P G 4 9	P R P G 4 8	P R P G 4 7	P R P G 4 6	P R P G 4 5	P R P G 4 4	P R P G 4 3	P R P G 4 2	P R P G 4 1	P R P G 4 0	P R P G 3 9	P R P G 3 8	P R P G 3 7	P R P G 3 6	P R P G 3 5	P R P G 3 4	P R P G 3 3	P R P G 3 2			

Table 1952. STCU2 register map (continued)

Addr. Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
0110h+ (NLBIST -1) * 40h	STCU_LB_MISREL	M I S R R E 3 1	M I S R R E 3 0	M I S R R E 2 9	M I S R R E 2 8	M I S R R E 2 7	M I S R R E 2 6	M I S R R E 2 5	M I S R R E 2 4	M I S R R E 2 3	M I S R R E 2 2	M I S R R E 2 1	M I S R R E 2 0	M I S R R E 1 9	M I S R R E 1 8	M I S R R E 1 7	M I S R R E 1 6	M I S R R E 1 5	M I S R R E 1 4	M I S R R E 1 3	M I S R R E 1 2	M I S R R E 1 1	M I S R R E 1 0	M I S R R E 9	M I S R R E 8	M I S R R E 7	M I S R R E 6	M I S R R E 5	M I S R R E 4	M I S R R E 3	M I S R R E 2	M I S R R E 1	M I S R R E 0				
0114h+ (NLBIST -1) * 40h	STCU_LB_MISREH	M I S R R E 6 3	M I S R R E 6 2	M I S R R E 6 1	M I S R R E 6 0	M I S R R E 5 9	M I S R R E 5 8	M I S R R E 5 7	M I S R R E 5 6	M I S R R E 5 5	M I S R R E 5 4	M I S R R E 5 3	M I S R R E 5 2	M I S R R E 5 1	M I S R R E 5 0	M I S R R E 4 9	M I S R R E 4 8	M I S R R E 4 7	M I S R R E 4 6	M I S R R E 4 5	M I S R R E 4 4	M I S R R E 4 3	M I S R R E 4 2	M I S R R E 4 1	M I S R R E 4 0	M I S R R E 3 9	M I S R R E 3 8	M I S R R E 3 7	M I S R R E 3 6	M I S R R E 3 5	M I S R R E 3 4	M I S R R E 3 3	M I S R R E 3 2				
0118h+ (NLBIST -1) * 40h	STCU_LB_MISRRL	M I S R R R 3 1	M I S R R R 3 0	M I S R R R 2 9	M I S R R R 2 8	M I S R R R 2 7	M I S R R R 2 6	M I S R R R 2 5	M I S R R R 2 4	M I S R R R 2 3	M I S R R R 2 2	M I S R R R 2 1	M I S R R R 2 0	M I S R R R 1 9	M I S R R R 1 8	M I S R R R 1 7	M I S R R R 1 6	M I S R R R 1 5	M I S R R R 1 4	M I S R R R 1 3	M I S R R R 1 2	M I S R R R 1 1	M I S R R R 1 0	M I S R R R 9	M I S R R R 8	M I S R R R 7	M I S R R R 6	M I S R R R 5	M I S R R R 4	M I S R R R 3	M I S R R R 2	M I S R R R 1	M I S R R R 0				
011Ch+ (NLBIST -1) * 40h	STCU_LB_MISRHH	M I S R R R 6 3	M I S R R R 6 2	M I S R R R 6 1	M I S R R R 6 0	M I S R R R 5 9	M I S R R R 5 8	M I S R R R 5 7	M I S R R R 5 6	M I S R R R 5 5	M I S R R R 5 4	M I S R R R 5 3	M I S R R R 5 2	M I S R R R 5 1	M I S R R R 5 0	M I S R R R 4 9	M I S R R R 4 8	M I S R R R 4 7	M I S R R R 4 6	M I S R R R 4 5	M I S R R R 4 4	M I S R R R 4 3	M I S R R R 4 2	M I S R R R 4 1	M I S R R R 4 0	M I S R R R 3 9	M I S R R R 3 8	M I S R R R 3 7	M I S R R R 3 6	M I S R R R 3 5	M I S R R R 3 4	M I S R R R 3 3	M I S R R R 3 2				
0120h+ (NLBIST -1) * 40h	STCU_LB_MISRELS W	M I S R R E S W 3 1	M I S R R E S W 3 0	M I S R R E S W 2 9	M I S R R E S W 2 8	M I S R R E S W 2 7	M I S R R E S W 2 6	M I S R R E S W 2 5	M I S R R E S W 2 4	M I S R R E S W 2 3	M I S R R E S W 2 2	M I S R R E S W 2 1	M I S R R E S W 2 0	M I S R R E S W 1 9	M I S R R E S W 1 8	M I S R R E S W 1 7	M I S R R E S W 1 6	M I S R R E S W 1 5	M I S R R E S W 1 4	M I S R R E S W 1 3	M I S R R E S W 1 2	M I S R R E S W 1 1	M I S R R E S W 1 0	M I S R R E S W 9	M I S R R E S W 8	M I S R R E S W 7	M I S R R E S W 6	M I S R R E S W 5	M I S R R E S W 4	M I S R R E S W 3	M I S R R E S W 2	M I S R R E S W 1	M I S R R E S W 0				
0124h+ (NLBIST -1) * 40h	STCU_LB_MISREH SW	M I S R R E S W 6 3	M I S R R E S W 6 2	M I S R R E S W 6 1	M I S R R E S W 6 0	M I S R R E S W 5 9	M I S R R E S W 5 8	M I S R R E S W 5 7	M I S R R E S W 5 6	M I S R R E S W 5 5	M I S R R E S W 5 4	M I S R R E S W 5 3	M I S R R E S W 5 2	M I S R R E S W 5 1	M I S R R E S W 5 0	M I S R R E S W 4 9	M I S R R E S W 4 8	M I S R R E S W 4 7	M I S R R E S W 4 6	M I S R R E S W 4 5	M I S R R E S W 4 4	M I S R R E S W 4 3	M I S R R E S W 4 2	M I S R R E S W 4 1	M I S R R E S W 4 0	M I S R R E S W 3 9	M I S R R E S W 3 8	M I S R R E S W 3 7	M I S R R E S W 3 6	M I S R R E S W 3 5	M I S R R E S W 3 4	M I S R R E S W 3 3	M I S R R E S W 3 2				
0128h+ (NLBIST -1) * 40h	STCU_LB_MISRRLS W	M I S R R R S W 3 1	M I S R R R S W 3 0	M I S R R R S W 2 9	M I S R R R S W 2 8	M I S R R R S W 2 7	M I S R R R S W 2 6	M I S R R R S W 2 5	M I S R R R S W 2 4	M I S R R R S W 2 3	M I S R R R S W 2 2	M I S R R R S W 2 1	M I S R R R S W 2 0	M I S R R R S W 1 9	M I S R R R S W 1 8	M I S R R R S W 1 7	M I S R R R S W 1 6	M I S R R R S W 1 5	M I S R R R S W 1 4	M I S R R R S W 1 3	M I S R R R S W 1 2	M I S R R R S W 1 1	M I S R R R S W 1 0	M I S R R R S W 9	M I S R R R S W 8	M I S R R R S W 7	M I S R R R S W 6	M I S R R R S W 5	M I S R R R S W 4	M I S R R R S W 3	M I S R R R S W 2	M I S R R R S W 1	M I S R R R S W 0				
012Ch+ (NLBIST -1) * 40h	STCU_LB_MISRHH SW	M I S R R R S W 6 3	M I S R R R S W 6 2	M I S R R R S W 6 1	M I S R R R S W 6 0	M I S R R R S W 5 9	M I S R R R S W 5 8	M I S R R R S W 5 7	M I S R R R S W 5 6	M I S R R R S W 5 5	M I S R R R S W 5 4	M I S R R R S W 5 3	M I S R R R S W 5 2	M I S R R R S W 5 1	M I S R R R S W 5 0	M I S R R R S W 4 9	M I S R R R S W 4 8	M I S R R R S W 4 7	M I S R R R S W 4 6	M I S R R R S W 4 5	M I S R R R S W 4 4	M I S R R R S W 4 3	M I S R R R S W 4 2	M I S R R R S W 4 1	M I S R R R S W 4 0	M I S R R R S W 3 9	M I S R R R S W 3 8	M I S R R R S W 3 7	M I S R R R S W 3 6	M I S R R R S W 3 5	M I S R R R S W 3 4	M I S R R R S W 3 3	M I S R R R S W 3 2				
Address space not used (available for future expansion)																																					
0600h+ (NMCUT -1) *4h	STCU_MB_CTRL	C S M	PTR									Reserved																									

82.7 Use cases and limitations

The STCU2 module is designed to give high flexibility during the on/off-line test. In the following are reported the programming operations that have to be performed to correctly use the functionality of this module considering the three typical usage cases:

- the Off-line Self Test Sequence after a reset trigger event
- the bypass mode of the Self Test Sequence after a reset trigger event
- the On-line Self-test Sequence

82.7.1 Off-Line Self-Test Sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and L/MBIST execution parameters stored into the NVM memory. The target is to cover the amount of physical defect into the digital logic and into the System RAMs/ROMs according to the System specifications. The sequence is the following:

- Unlock the Off-Line STCU2 access writing the Key1/Key2 sequence into the STCU_SKC register
- Program the LBIST fault reaction conditions (UF/RF) setting the STCU_LBUFM register
- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU_MBUFML/M/H registers
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU_ERR_FM register
- Program the STCU_LB_CTRL, STCU_LB_PCS, STCU_MISREL/H registers of each LBIST to be executed. Optionally, also the STCU_LB_PRPGL/H can be programmed in order to load a specific seed.
- Program the STCU_MB_CTRL registers of each NMCUT to be executed
- Program the STCU_WDG register to define the time budget assigned for the L/MBIST execution
- Program the STCU_CFG register in order to define: the core and L/MBIST TCK clock prescaling factor setting the CLK_CFG bits; the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase; the PMOSEN or CHKBRD in order to define the

algorithm to be used during MBIST run, the CRCREN bit to enable the self-checking feature and finally set the pointer to the first LBIST/NMCUT to be executed.

- In case the internal CRC comparison has been enabled, program the CRCE value expected at the end of the off-line Self Test sequence into the STCU_CRCE register
- In case the PLL usage has been enabled, program the STCU_PLL_CFG value in order to set the required system frequency
- Program the RUN bit into the STCU_RUN register to enable the Off-Line Self-Test execution and the L/MBPLEN bits to enable the PLL during L/MBIST execution according to STCU_PLL_CFG register content
- Start the Off-Line Self-Test and, depending on PLL management and L/MBIST execution order, .
- Wait the WDGE0C run time execution specified into the STCU_WDG register.
- The STCU2 switch-off the core clock at the end of the Self-Test run and release the PLL control in case this feature has been enabled.
- The STCU2 reset the STCU_RUN[RUN] bit and the device exit from boot sequence. In case there are failures, the STCU2 flag these failures toward the FCCU rising one/both the *stcu_ncf* and *stcu_cf* lines.
- The software application read the STCU_LBS flag register to check the failed LBIST when UF or RF conditions is/are detected.
- The software application read the STCU_LBE flag register to check the LBIST still on-going when UF or RF is/are detected.
- The software application read the STCU_MBSL/M/H flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
- The software application read the STCU_MBEL/M/H flag registers, depending on the number of memories, to check the RAMs/ROMs Memory BIST still on-going when UF or RF is/are detected.
- The software application read the bit INV, ENGE, CRCS, WDTO, LOCKE of the STCU_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.
- In case the CRCEN bit has been enabled, the software application read the CRCE and CRCR registers to check the coherence with the bit CRCS of the STCU_ERR register.
- The software application read for each device's LBIST, the STCU_LBMISREL/H and STCU_LBIST_MISRRL/H registers to check the coherence with the STCU_LBS bits.

Note: :The software application read operations on the STCU2 CRC and LBIST MISR registers after off-line self Test execution has been successfully performed are just an additional safety layer to be added to improve the already implemented Auto Self-Test feature included into STCU2.

82.7.2 On-Line Self-Test Sequence

This is the typical mode of using the STCU2 module during the application run. The IPS interface is used to program the STCU2 registers and to schedule the L/MBIST execution. Since the On-Line L/MBIST operations clean-up the content of the related Digital Logic

(LBIST) or RAM (MBIST) the activation of this mode has to be done carefully. The suggested sequence is the following:

- Unlock the On-Line STCU2 access writing the Key1/Key2 sequence into the STCU_SKC register
- Check and/or clean-up the bit WRP into the STCU_CFG register to open the IPS access on the STCU2 On-Line Self-Test registers
- The LBIST, MBIST and STCU2 internal fault reaction should have been already programmed during the Off-Line Self-Test sequence and so they should not be programmed again. However, in case they have not been programmed or there is the need of changing them for any reason, they can be overwritten setting respectively the registers: STCU_LBUFM, STCU_MBUFML/M/H and STCU_ERR_FM.
- Program the Reset management during on-Line LBIST execution setting the STCU_LBRMSW register
- The STCU_LB_CTRL, STCU_LB_PCS should have been already programmed during Off-Line Self-test sequence and so they should not be programmed again. However, in case they have not been programmed or there is the need of changing them for any reason, they can be overwritten. The registers STCU_LB_MISRELSW/HSW have to be programmed according to the expected signature.
- Overwrite/Program the STCU_MB_CTRL registers of each NMCUT to be executed
- Overwrite/Program the STCU_WDG register to define the time budget assigned for the L/MBIST execution
- Overwrite/Program the STCU_CFG register in order to define: the core and L/MBIST TCK clock prescaling factor setting the CLK_CFG bits; the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase; the PMOSEN or CHKBRD in order to define the algorithm to be used during MBIST run, the CRCREN bit to enable the self-checking feature and finally set the pointer to the first LBIST/NMCUT to be executed.
- In case the internal CRC comparison has been enabled, overwrite/program the CRCE value expected at the end of the on-line Self Test sequence into the STCU_CRCE register
- Program the RUNSW bit into the STCU_RUNSW register to enable the On-Line Self-Test execution, the L/MBIE to enable the Interrupt requests, and the L/MBSWPLEN bits to enable the PLL lock signal monitor during L/MBIST execution.
- Start the On-Line Self-Test and depending on PLL management and L/MBIST execution order.
- Wait the WDGEOC run time execution specified into the STCU_WDG register
- In case the L/MBIE have been enabled, an interrupt is generated at the end of all the L/MBIST execution. The software application has to manage these interrupt signals and clean-up the related flag (L/MBIFLG) into the STCU_INT_FLG register.
- The STCU2 switch-off the core clock at the end of the Self-Test run
- The functional reset activated is managed by the STCU_LBRMSW.
- In case, at least one of the selected LBIST enables the Global functional reset, the STCU2 requests a functional reset which restarts the system while, when all the

selected LBIST enables its own dedicated functional reset, only these local reset lines are applied and only the related LBIST partitions are cleaned-up.

- The STCU2 resets the STCU_RUNSW[RUNSW] bit. In both the reset cases and in case of error/s, the STCU2 flags the failure toward the FCCU rising one/both of the *stcu_ncf* and *stcu_cf* lines.
- The software application read the STCU_LBSSW flag register to check the failed LBIST when UF or RF conditions is/are detected
- The software application read the STCU_LBESW flag register to check the LBIST still on-going when UF or RF is/are detected
- The software application read the STCU_MBSLSW/MSW/HSW flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected
- The software application read the STCU_MBELSW/MSW/HSW flag registers, depending on the number of memories, to check the RAMs/ROMs Memory BIST still on-going when UF or RF is/are detected
- The software application read the bit INVPSW, ENGESW, CRCSSW, WDTOSW, LOCKESW of the STCU_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected
- In case the CRCEN bit has been enabled, the software application read the CRCE and CRCE registers to check the coherence with the bit CRCSSW of the STCU_ERR register
- The software application read for each device's LBIST, the STCU_LBMISRELSW/HSW and STCU_LBIST_MISRRLSW/HSW registers to check the coherence with the STCU_LBS bits

Note: The software application read operations on the STCU2 CRC and LBIST MISR registers after on-line self-test execution has been successfully performed are just an additional safety layer to be added to improve the already implemented Auto Self-Test feature included in the STCU2.

82.7.3 Bypass USER Mode

In this case, the USER application parameters stored in FLASH and read by SSCM are written in order to skip the Self-Test procedure after a reset trigger event is applied. It might be useful in case the device is not configured for Safety application. The sequence is the following:

1. Unlock the STCU2 access writing the Off-Line Key1/Key2 sequence into the STCU_SKC register
2. Set the BYP bit in the STCU_RUN register

After BYP bit is set, the core clock is switched off, the Self-Test sequence is not applied and the system can wake up and start the user application.

82.7.4 Design implementation information

The following list reports the limitations related to the current implementation:

- Every NMCUT MBIST or LBIST can be run one time during the On/Off-line Self-test procedure.
- It is not possible to run LBIST and MBIST concurrently.
- Since during the On-Line Self-Test the software has the full control of the system while applications are running, the STCU2 is not allowed to take the control of the PLL but can just monitor the lock signal to flag wrong unlock events.

82.8 Open issues

No open issues.

83 Register Protection (REG_PROT)

83.1 Introduction

83.1.1 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

Note: Please see the device configuration chapter of the device reference manual for details of the protected registers and bytes.

The protection module is located between the module under protection and the AIPS. This is shown in [Figure 2093](#).

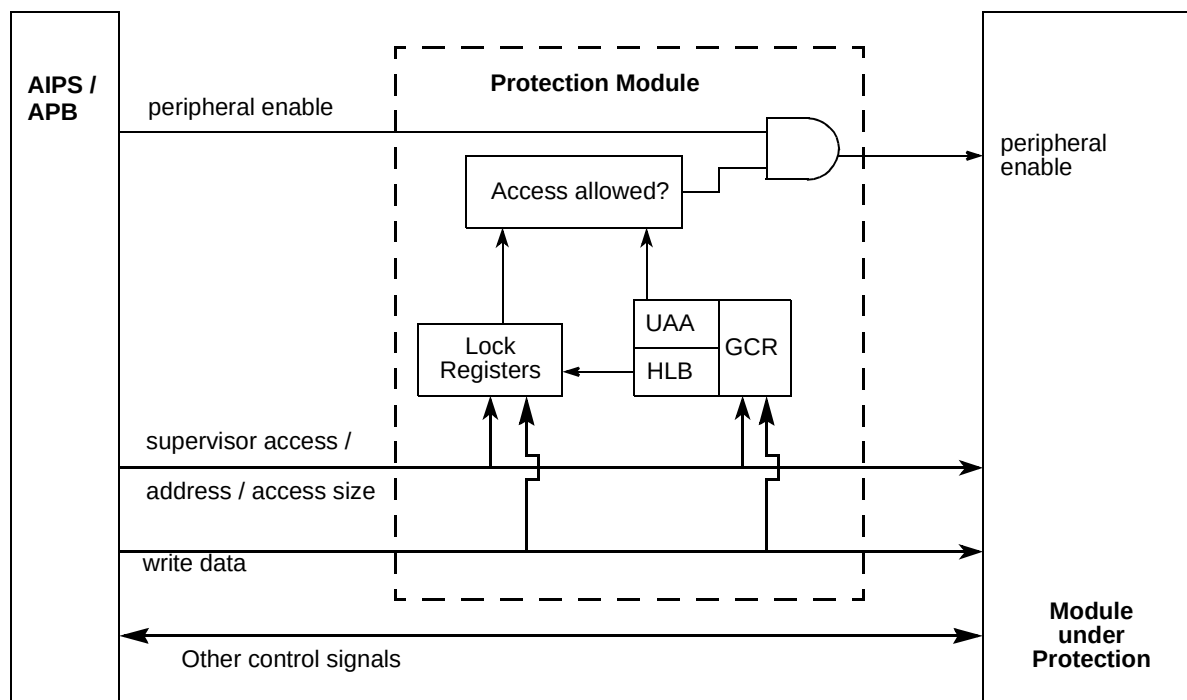


Figure 2093. REG_PROT Block Diagram

83.1.2 Features

The Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

Also, hard lock bit protection can only be cleared by a system reset once set

83.1.3 Modes of operation

The Register Protection module is operable when the module under protection is operable.

83.2 External signal description

There are no external signals.

83.3 Memory map and register definition

This section provides a detailed description of the memory map of a module using the REG_PROT. The original 16 KB module memory space is divided into five areas as shown in [Figure 2094](#).

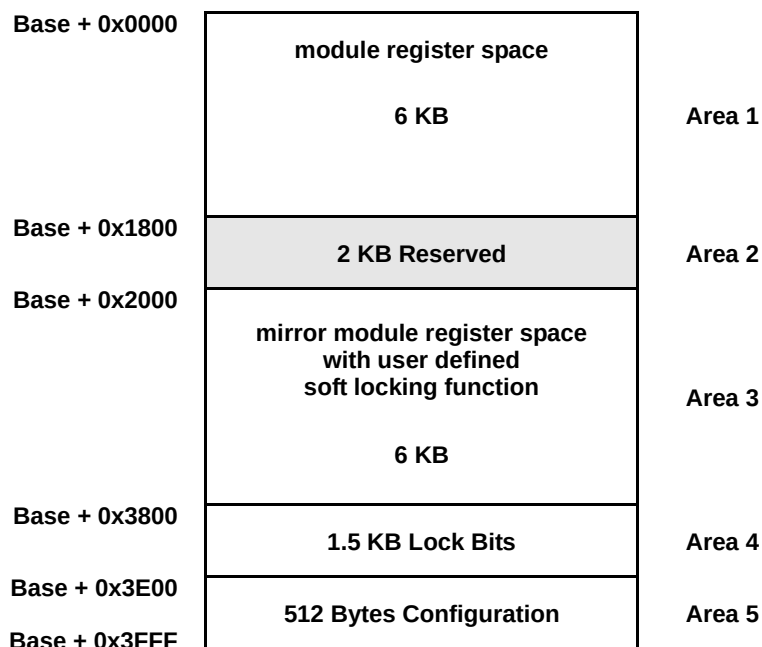


Figure 2094. REG_PROT Memory Diagram

- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an atomic write and lock operation. Not all registers in area 1 need to have protection defined by associated

Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.

- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-Bit registers in Areas 4 and 5 will result in a transfer error.

83.3.1 Memory map

[Figure 1953](#) shows the memory map for a module which is protected via a REG_PROT module.

Table 1953. REG_PROT Memory Map

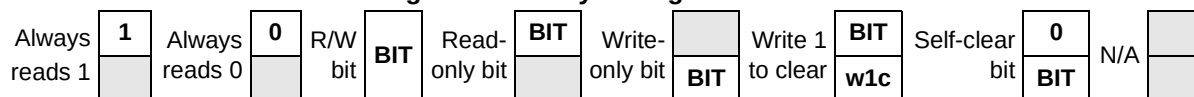
Offset	Use	Location
0x0000	Module Register 0 (MR0)	on page 3601
0x0001	Module Register 1 (MR1)	on page 3601
0x0002	Module Register 2 (MR2)	on page 3601
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)	on page 3601
0x1800 - 0x1FFF	Reserved	
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)	on page 3601
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)	on page 3601
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)	on page 3601
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3	on page 3601
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7	on page 3601
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 - Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143	on page 3601
0x3E00 - 0x3FFB	Reserved	
0x3FFC	Global Configuration Register (GCR)	on page 3603

Note: *Reserved registers in area #1 will be handled as specified in the protected module's documentation.*

83.3.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Figure 2095. Key to Register Fields



83.3.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

83.3.2.2 Module Register and Set Soft Lock Bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBR n .SLB m , according to the mapping described in [Table 1954](#).

83.3.2.3 Soft Lock Bit Register (SLBR0-1535)

These registers hold the Soft Lock Bits for the protected registers in memory area #1.

Figure 2096. Soft Lock Bit Register (SLBR n)

Address 0x3800-0x3DFF					Access: Read always Supervisor write			
	0	1	2	3	4	5	6	7
R	0	0	0	0	SLB0	SLB1	SLB2	SLB3
W	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

Table 1954. SLBR n field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB): WE0 enables writing to SLB0 1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB): WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified

Table 1954. SLBR_n field descriptions (continued)

Field	Description
2 WE2	Write Enable Bits for Soft Lock Bits (SLB): WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB): WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MR _n register: SLB0 can block accesses to MR[_n * 4 + 0] 1 Associated MR _n byte is locked against write accesses 0 Associated MR _n byte is unprotected and writeable
5 SLB1	Soft Lock Bits for one MR _n register: SLB1 can block accesses to MR[_n * 4 + 1] 1 Associated MR _n byte is locked against write accesses 0 Associated MR _n byte is unprotected and writeable
6 SLB2	Soft Lock Bits for one MR _n register: SLB2 can block accesses to MR[_n * 4 + 2] 1 Associated MR _n byte is locked against write accesses 0 Associated MR _n byte is unprotected and writeable
7 SLB3	Soft Lock Bits for one MR _n register: SLB3 can block accesses to MR[_n * 4 + 3] 1 Associated MR _n byte is locked against write accesses 0 Associated MR _n byte is unprotected and writeable

Table 1955 gives some examples how SLBR_n.SLB and MR_n go together:

Table 1955. Soft Lock Bits vs. Protected Address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

83.3.2.4 Global Configuration Register (GCR)

This register controls the configuration of the locking of the SLB's and the level of user access allowed to the protected registers.

Figure 2097. Global Configuration Register (GCR)

Offset 0x3FFC																Access: Read Always Supervisor write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	HLB	0	0	0	0	0	0	0	UAA	0	0	0	0	0	0	0															
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1956. GCR field descriptions

Field	Description
HLB	Hard Lock Bit. This register can not be cleared once it is set by software. It can only be cleared by a system reset. 1 All SLB bits are write protected and can not be modified 0 All SLB bits are accessible and can be modified.
UAA	User Access Allowed. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.

Note: The GCR.UAA bit has no effect on the allowed access modes for the registers in the REG_PROT module.

83.4 Functional description

83.4.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size depend on the SoC and/or module..

For all addresses that are protected there are $SLBRn.SLBm$ bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding $SLBRn.SLBm$ bit is always 0b0 no matter what software is writing to.

83.4.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding $SLBRn.SLBm$ bit needs to be changed. This can be done using the following methods:

- Modify the $SLBRn.SLBm$ directly by writing to area #4
- Set the $SLBRn.SLBm$ bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

83.4.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each $SLBRn.SLBm$ bit has a mask bit $SLBRn.WEm$ which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

Figure 2098 shows two modification examples. In the left example there is a write access to the $SLBRn$ register specifying a mask value which allows modification of all $SLBRn.SLBm$ bits. The example on the right specifies a mask which only allows modification of the bits $SLBRn.SLB[3:1]$.

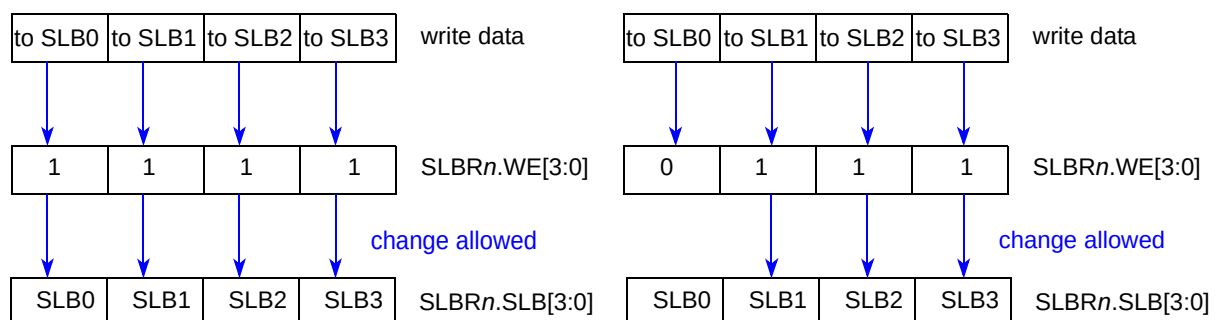


Figure 2098. Change lock settings directly via Area #4

Figure 2098 showed four registers that can be protected 8-bit wise. In *Figure 2099* registers with 16-bit protection and in *Figure 2100* registers with 32-bit protection are shown:

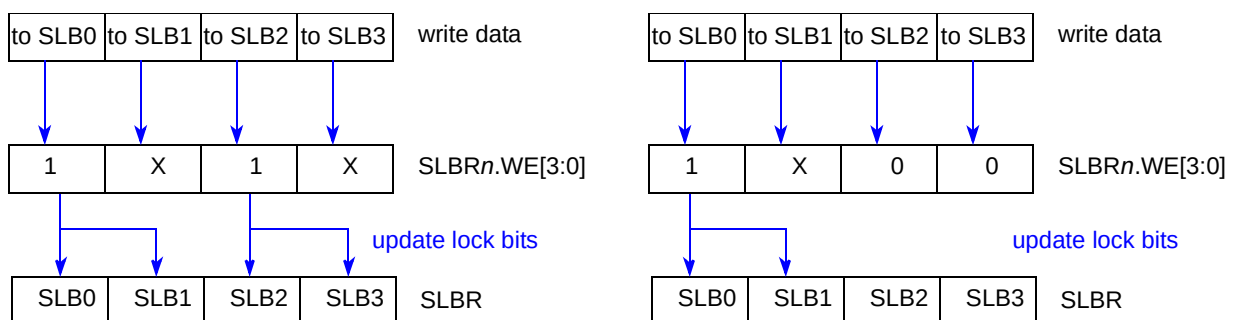
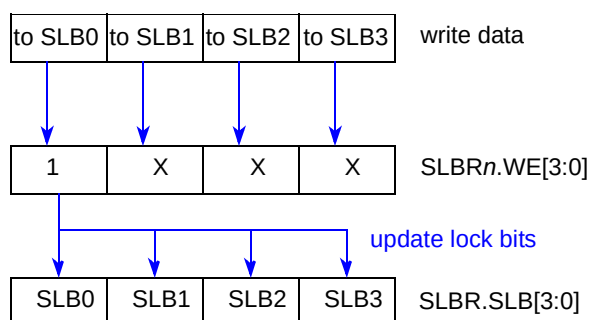


Figure 2099. Change lock settings for 16-bit protected addresses

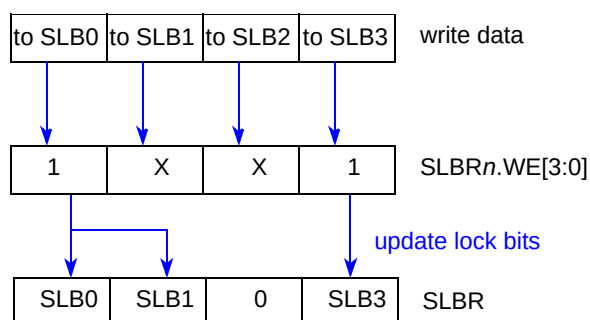
On the right side of [Figure 2099](#) it is shown that the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[1] also. This is done as the address reflected by SLBRn.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBRn.WE[0] must be set while SLBRn.WE[1] does not matter. As the enable bits SLBRn.WE[3:2] are cleared the lock bits SLBRn.SLB[3:2] remain unchanged.

In the example on the left side of [Figure 2099](#) the data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] and the data written to SLBRn.SLB[2] is mirrored to SLBRn.SLB[3] as for both registers the write enables are set.

In [Figure 2100](#) a 32-bit wise protected register is shown. When SLBRn.WE[0] is set the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[3:1] also. Otherwise SLBRn.SLB[3:0] remains unchanged.

**Figure 2100. Change lock settings for 32-bit protected addresses**

In [Figure 2101](#) an example is shown which has a mixed protection size configuration:

**Figure 2101. Change lock settings for mixed protection**

The data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBRn.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBRn.SLB[3] is written to SLBRn.SLB[3].

83.4.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. [Figure 2102](#) shows one example:

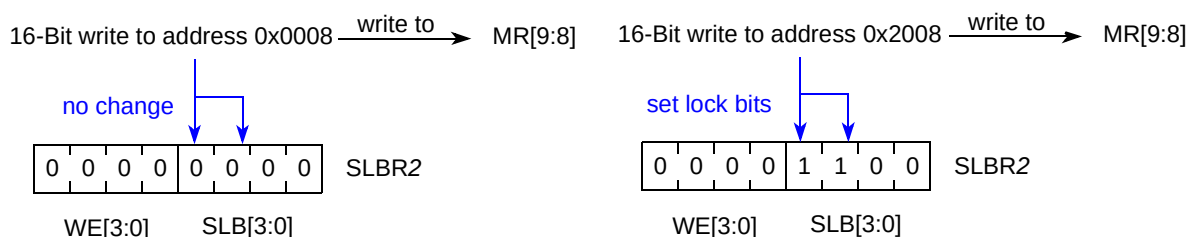


Figure 2102. Enable Locking Via Mirror Module Space (Area #3)

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of [Figure 2099](#)).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of [Figure 2099](#)).

[Figure 2103](#) shows an example where some addresses are protected and some are not:

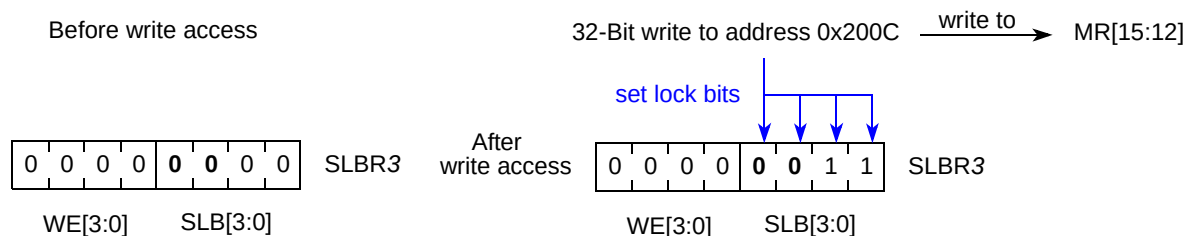


Figure 2103. Enable locking for protected and unprotected addresses

In the example in [Figure 2103](#) addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note: Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

83.4.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in [Section 83.4.2.1, Change lock settings directly via Area #4](#) and [Section 83.4.2.2, Enable locking via mirror module space \(Area #3\)](#) is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there was a system reset.

83.4.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to [Figure 2094](#)

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set

83.5 Initialization/application information

83.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See [Section 83.3.2, Register descriptions](#)). In summary, after reset, locking for all MR_n registers is disabled. The registers can be accessed in Supervisor Mode only.

83.5.2 Writing C code using the register protection scheme

There is a set of macros provided as part of the device specific header file, which defines the memory map, the peripheral registers, and the fields of these registers. These macros are intended to make working with the register protection scheme easier for the developers of device driver software and/or other application code. This section describes these macros and how to use them.

A first macro is made available to perform a write to the mirrored module register space (Area 3). As described in this document, this results in concurrently setting the corresponding soft lock bit, while writing to the module register. There are three flavors of this macro, to account for the different size of the related module register `<thereg>` (the value of `<size>` is either 8, 16, or 32 bit):

```
WRITE_WITH_LOCK<size>(<thereg>, <newvalue>)
```

This macro writes the value `<newvalue>` into the register `<thereg>` assuming a register size of `<size>`. The parameter `<thereg>` must be the name of a register using the notation in the device specific header file.

A second set of macros is made available to work with the soft lock bits provided by the register protection scheme. The value of these bits associated with a particular module register `<thereg>` can be retrieved, set, and cleared. The macros provided for this purpose are:

```
GET_SOFTLOCK(<thereg>, <dest>)
SET_SOFTLOCK<size>(<thereg>)
CLR_SOFTLOCK<size>(<thereg>)
```

The macro `GET_SOFTLOCK` retrieves the value of the soft lock bit register associated with the given register `<thereg>` and stores it in the variable `<dest>`; which is always an eight bit

value. The other two macros (SET_SOFTLOCK, CLR_SOFTLOCK) set or clear the softlock bits associated with the given register <thereg>; assuming this register has a size of 8, 16, or 32 bit as indicated by the macro name. For all three macros, the parameter <thereg> must be the name of a register using the notation in the device specific header file.

Three more macros are made available to modify bits in the Global Configuration Register (GCR) of the protection gasket for the corresponding block. In contrast to the earlier ones, these macros receive the base address of the corresponding block (usually named <block_name>_BASEADDRESS) as a parameter.

SET_HARDLOCK(base)

Sets the Hard Lock Bit HLB in the GCR register associated with the module identified by the given base address <base>.

USER_ACCESS_FORBIDDEN(base)

Clears the User Access Allowed Bit UAA in the GCR register associated with the module identified by the given base address <base>. When cleared, this bit denies any write access to the protected module in user mode and generates a transfer error in case of an attempt to write a protected register.

USER_ACCESS_ALLOWED(base)

Sets the User Access Allowed Bit UAA in the GCR register associated with the module identified by the given base address <base>. When set, this bit permits write accesses to the protected module in user mode when the corresponding register are not additionally protected by other means.

Finally, two more macros are provided to permit direct access to the registers in the register protection gasket using the same semantic as other register definitions in the device specific header file:

LOCK_SLB(thereg)

provides the content of the soft lock bit register associated with the register <thereg>; the parameter <thereg> must be the name of the corresponding module register using the notation in the device specific header file.

LOCK_GCR(base)

provides the content of the Global Configuration Register GCR for the block identified by its base address <base>; the parameter <base> must be the base address of the corresponding block.

84 Password and Device Security Module (PASS)

84.1 Introduction

84.1.1 Overview

The Password and device security (PASS) module shown in [Figure 2104](#), receives password challenges and determines their validity. It also maintains the device security and access states.

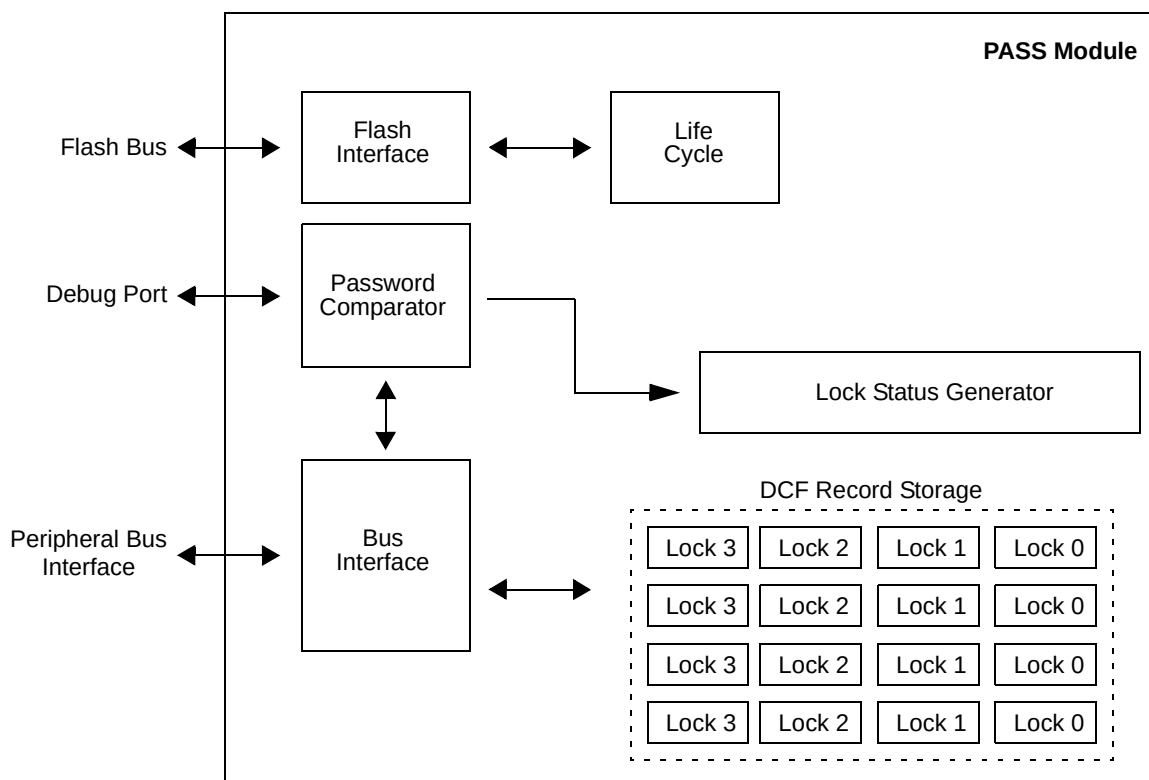


Figure 2104. PASS module block diagram

84.1.2 Features

The PASS includes these distinctive features:

- Life cycle status
- Password comparison
- JTAG password comparison
- Production Disable control

84.1.3 Modes of operation

The PASS operates identically in all system modes.

84.2 External signal description

The PASS has 1 external pin that is used for Production Disable control.

84.3 Memory map/register definition

This section provides a detailed description of all memory-mapped registers and DCF clients in the PASS.

Where a DCF client is listed, the reset value of that register maybe automatically preloaded during reset based on the data stored in DCF Records in flash UTEST region.

[Table 1957](#) shows the memory map for the PASS. Note that all addresses are offsets; the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Chip Select (CS[14:0]) for the PASS module as defined in the DCF Record Section of the Reference Manual

Table 1957. Module memory map

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section/Page
0x0000	Life Cycle Status Register (LCSTAT)	R	0x0000_00xx ⁽¹⁾	0x00A0	84.3.1.1/3611
0x0008	Challenge Selector Register (CHSEL)	R/W	0x0000_0000	None	84.3.1.2/3613
0x0010	Challenge Status Register (CSTAT)	R		None	84.3.1.3/3614
0x0020	Challenge Input Register 0 (CIN0)	W	0x0000_0000	None	84.3.1.4/3615
0x0024	Challenge Input Register 1 (CIN1)	W	0x0000_0000	None	84.3.1.4/3615
0x0028	Challenge Input Register 2 (CIN2)	W	0x0000_0000	None	84.3.1.4/3615
0x002C	Challenge Input Register 3 (CIN3)	W	0x0000_0000	None	84.3.1.4/3615
0x0030	Challenge Input Register 4 (CIN4)	W	0x0000_0000	None	84.3.1.4/3615
0x0034	Challenge Input Register 5 (CIN5)	W	0x0000_0000	None	84.3.1.4/3615
0x0038	Challenge Input Register 6 (CIN6)	W	0x0000_0000	None	84.3.1.4/3615
0x003C	Challenge Input Register 7 (CIN7)	W	0x0000_0000	None	84.3.1.4/3615
Censorship					
None	Censorship	None	0x0000_0000	0x00B0	84.5.4/3629
Production Disable					
None	Production Disable	None	0x0000_0000	0x00C0	84.3.1.6/3616
Clock Jitter Enable					
0x00D0	Clock Jitter Enable	R/W	0xFFFF_FFFF	None	84.3.1.7/3617
Password Group 0					
0x0100	Lock 0 Status Register (LOCK0_PG0)	R/W	0xxxxx_xxxx ⁽²⁾	0x0100	84.3.1.8/3618
0x0104	Lock 1 Status Register (LOCK1_PG0)	R/W	0xx000_xxxx	0x0104	84.3.1.9/3620
0x0108	Lock 2 Status Register (LOCK2_PG0)	R/W	0xxxxx_xxxx ⁽²⁾	0x0108	84.3.1.10/3621
0x010C	Lock 3 Status Register (LOCK3_PG0)	R/W	0xxxxx_xxxx ⁽²⁾	0x010C	84.3.1.11/3623

Table 1957. Module memory map (continued)

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section/Page
Password Group 1					
0x0110	Lock 0 Status Register (LOCK0_PG1)	R/W	0xxxxx_xxxx ⁽²⁾	0x0110	84.3.1.8/3618
0x0114	Lock 1 Status Register (LOCK1_PG1)	R/W	0xx000_xxxx	0x0114	84.3.1.9/3620
0x0118	Lock 2 Status Register (LOCK2_PG1)	R/W	0xxxxx_xxxx ⁽²⁾	0x0118	84.3.1.10/3621
0x011C	Lock 3 Status Register (LOCK3_PG1)	R/W	0xx0xx_xxxx ⁽²⁾	0x011C	84.3.1.11/3623
Password Group 2					
0x0120	Lock 0 Status Register (LOCK0_PG2)	R/W	0xxxxx_xxxx ⁽²⁾	0x0120	84.3.1.8/3618
0x0124	Lock 1 Status Register (LOCK1_PG2)	R/W	0xx000_xxxx	0x0124	84.3.1.9/3620
0x0128	Lock 2 Status Register (LOCK2_PG2)	R/W	0xxxxx_xxxx ⁽²⁾	0x0128	84.3.1.10/3621
0x012C	Lock 3 Status Register (LOCK3_PG2)	R/W	0xx0xx_xxxx ⁽²⁾	0x012C	84.3.1.11/3623
Password Group 3					
0x0130	Lock 0 Status Register (LOCK0_PG3)	R/W	0xxxxx_xxxx ⁽²⁾	0x0130	84.3.1.8/3618
0x0134	Lock 1 Status Register (LOCK1_PG3)	R/W	0xx000_xxxx	0x0134	84.3.1.9/3620
0x0138	Lock 2 Status Register (LOCK2_PG3)	R/W	0xxxxx_xxxx ⁽²⁾	0x0138	84.3.1.10/3621
0x013C	Lock 3 Status Register (LOCK3_PG3)	R/W	0xx0xx_xxxx ⁽²⁾	0x013C	84.3.1.11/3623

1. Reset value depends on life cycle status

2. Reset value depends on password group lock state

Note: Non-implemented registers generate bus aborts, if enabled by the SOC.

84.3.1 Register descriptions

The following registers are available in the PASS. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits should only be written with '0'. They will always read back '0'.

84.3.1.1 Life Cycle Status register (LCSTAT)

The Life Cycle Status Register reflects the production status and censorship status of the device. It's possible to mature the device (move the device life cycle forward), but never to make the device younger (move the life cycle to an earlier state).

The reset value for the LCSTAT register LIFE field is defined by a DCF client that is automatically pre-loaded during reset from data in flash UTEST region. The DCF client is 0xFF by default and configured such that bits can only be written from '1' to '0'. Subsequent DCF records can move the LIFE field through the Life Cycle states.

Figure 2105. Life Cycle Status register (LCSTAT)

Note: 'X' means the reset value is coming from a DCF client

Offset 0x0000

Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LIFE							
W																
Reset	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x

PASS_LCSTAT register functions are as shown in [Table 1958](#).

Table 1958. LCSTAT register field descriptions

Field	Description
0 CNS	This field indicates the System Censoring Status. The bit can be set (or cleared) via DCF Record in UTest Flash. See section Section 84.5.4, "Censoring/uncensoring the device" . 1 Device is censored 0 Device is uncensored
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to 1, otherwise it is 0. Unsuccessful transmission clears this bit.
2–23	Reserved
24–31 LIFE	This field shows the value of the life cycle DCF Record. The field can not be written directly, but is determined by the Life cycle records which have been written into the UTest Flash. 1111_110- Customer Delivery 1111_10-- OEM Production 1111_0--- In Field ---0---- Failure Analysis mode ('-' indicates that the value of the bit is not considered for interpretation of the life cycle.)

The effects of the life cycle on security is described in [Section 84.4.1, Life cycle and security](#).

84.3.1.2 Challenge Selector register (CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

Figure 2106. CHSEL register

Offset 0x0008												Access: Read/Write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GRP	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CHSEL register functions are as shown in [Table 1959](#).

Table 1959. CHSEL register field descriptions

Field	Description
0–29	Reserved
30–31 GRP	Password Group to unlock 00Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

When a master accesses this register, the PASS module will store the number of the master in the LOCK3_PGN register of the selected passgroup and set the PGL bit in that register.

The ID of the accessing master will set the CMST field of the CSTAT register.

84.3.1.3 Challenge Status register (CSTAT)

This register provides status information for the password challenge.

Figure 2107. CHSTAT register

Offset 0x0010												Access: Read				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CMST			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CHSTAT register functions are as shown in [Table 1960](#).

Table 1960. CHSEL register field descriptions

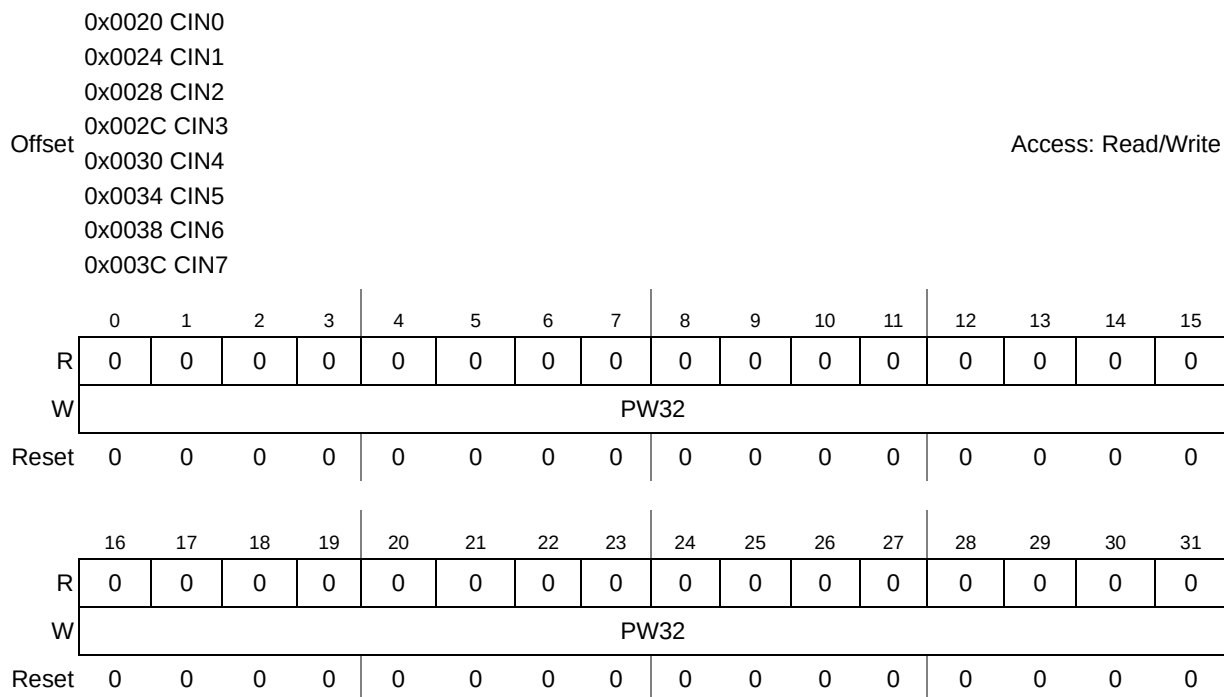
Field	Description
0–27	Reserved
28–31 CMST	ID of the master which has last written the CHSEL register

Only the master shown in this register can access the CINn registers—access by other masters will result in a comparison fail, regardless of the values written.

84.3.1.4 Challenge Input register (CIN_n)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

Figure 2108. CIN_n register



CIN register functions are as shown in [Table 1961](#).

Table 1961. CIN register field descriptions

Field	Description
0–31 PW32	32 bits of the 256 bit password challenge. Only 32-bit writes may be used.

After programming the Challenge Selector Register the password challenge needs to be written with eight 32bit writes to CIN0 to CIN7 - in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written the password will be compared. The write access will only complete once the comparison has been executed.

All accesses to these registers need to be made by the bus master which wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison will fail.

84.3.1.5 Censorship

Censorship is a DCF client that is NOT accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the flash UTEST region. The Censorship DCF clients defines if the MCU is censored.

DCF Client Address: 0x00B0 Censorship

Access: DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not defined
2. This value is loaded by DCF Record during reset

Table vi

Field	Description
0–15	Reserved
16:31	Censorship If this value is 0x55AA the MCU is NOT censored If this value is anything other than 0x55AA, the MCU is censored

84.3.1.6 Production Disable

Production Disable is a DCF client that is NOT accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the flash UTEST region. The Production disable DCF clients defines two different aspects of the Production Disable feature:- Debug Disable; and Hardware Interlock.

The DCF client implements 2 bits. Their default values are both '0'. The DCF client can only be written once. Writes from subsequent DCF records will be ignored.

For more information on Production disable, see [Section 84.5.6, Production disable](#)

84.3.1.6.1 Debug Disable (DD)

Debug Disable is an optional feature that, once the Lifecycle is >= In Field, overrides all other debug controls, and disables the debug interface. The Debug Disable feature has higher priority than Censorship, JTAG password and PASS_LOCK3[DBL] bit. All Debug access is blocked while Clock Jitter is NOT enabled.

Once Clock Jitter IS enabled, the Debug interface is NOT disabled and can be enabled temporarily or permanently using Censorship or JTAG password or PASS_LOCK3[DBL] bit.

84.3.1.6.2 Hardware Interlock (HI)

Hardware Interlock is an optional feature that, once the Lifecycle is >= In Field, blocks programming of the BAF flash block while an external hardware pin is asserted low. This feature has higher priority than PASS_LOCK0[BAF] bits. While the pin is asserted, no software action anywhere in the MCU can cause the BAF to be programmed.

Once the hardware pin is asserted high, programming of BAF is controlled by PASS_LOCK0[BAF] bits.

Figure 2109. Production Disable DCF client definition

DCF Client Address: 0x00C0 Production Disable

Access: DCF Record Write Once Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	DD ⁽¹⁾	HI ⁽¹⁾														
Reset	0	0	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾

1. This value is loaded by DCF Record during reset
2. This bit is not defined

Table 1962. Production Disable DCF Client bit definition

Field	Description
0 DD	Debug disable. '0' = Debug Interface is Enabled and Disabled using Censorship, JTAG password & PASS_LOCK3[DBL] bit as described elsewhere '1' = Debug Interface is disabled while Clock Jitter is NOT enabled
1 HI	Hardware Interlock '0' = BAF flash block programming is controlled by PASS_LOCK0[BAF] bits '1' = BAF flash block is locked for programming while external hardware pin is asserted '0'
2–31	Reserved

84.3.1.7 Clock Jitter Enable

Software accessible one time writable register. This register stores the jitter enable status for Pseudo Random Clock Divider Module (PRCD). The default value of this register is '1'. When this bit is cleared, the PRCD is enabled and the baud rate clock to several serial communications modules will include large amounts of pseudo random clock edge jitter, making the serial communications modules unusable. For details about which serial communications modules are affected, see the clocking section of the Reference Manual.

For more information on Production disable, see [Section 84.5.6, Production disable](#)

When Production disable bit[0] (Gated Debug Enable) is asserted, Chip is in Field and CAN Jitter is not enabled then debug access is blocked.

Figure 2110. Clock Jitter Enable

Offset 0x00D0 Clock Jitter Enable

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CJE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 1963. Production Disable DCF Client bit definition

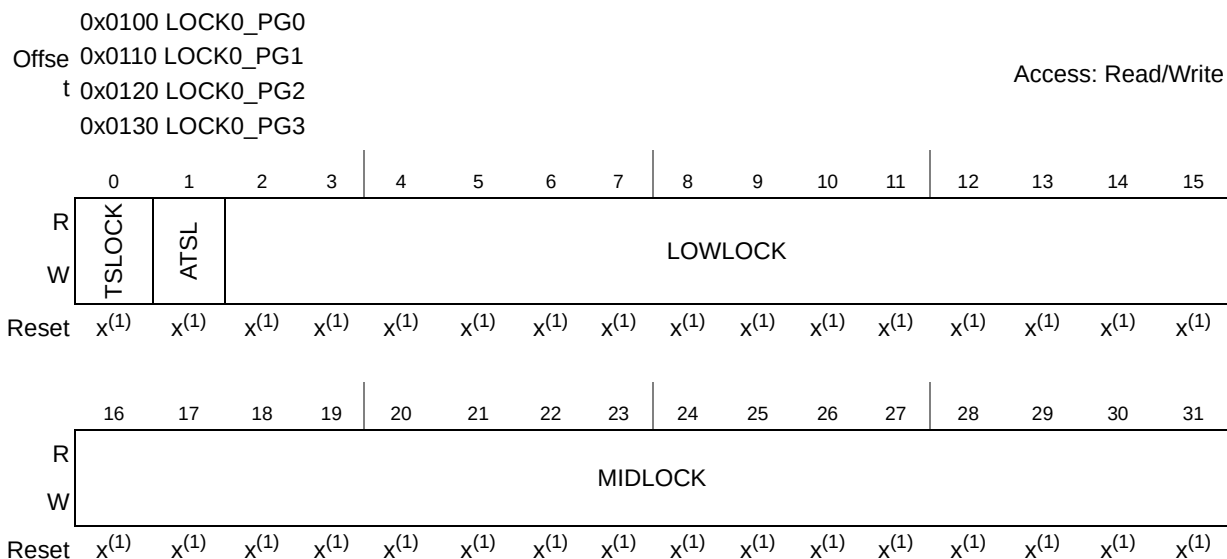
Field	Description
0 CJE	Clock Jitter Enable. 0 Clock Jitter is enabled. The baud rate clock to Serial communications modules includes large amounts of pseudo random clock edge jitter 1 Clock Jitter is disabled. The baud rate clocks function normally
1–31	Reserved

84.3.1.8 Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register will become writable. For additional details of the LOCK bits, refer to the flash memory description.

The LOCK bits will not come into effect before the life cycle is matured to “OEM Production” or older.

Figure 2111. LOCK0_PGn register



1. If no DCF records have been written to change the lock state, the bit will reset to '1'.

LOCK0_PGn register functions are as shown in [Table 1964](#).

Table 1964. LOCK0_PGn register field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable).
1 ATSL	Alternate Interface UTest NVM Lock. This bit is used to lock the UTest NVM block from programs on the alternate interface (erase protection not needed since UTest NVM is OTP and not erasable).
2–15 LOWLOCK	Low Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16–31 MIDLOCK	Mid Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTest Flash block and copied over during reset. The default value before any DCF record is written, is '1' which indicates a block is locked.

During the life cycle “Customer Delivery”, the user must program a valid password or a DCF record that write this register to unprotect the UTest for each password group to prevent permanent and irreversible locking of flash blocks.

Warning: Failure to program valid passwords or DCF records for all password groups before the life cycle is matured to “OEM Production” will render the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible.

However unless the life cycle is older than “OEM Production”, the Flash blocks will always be unlocked, regardless of the values in the password groups.

When Production Disable register bit[1](BAF PLock Hardware Interlock enable) is asserted and lifecycle is infield, then the external signal will be ORed with BAF_plock(low_lock[0]) of LOCK0_PGx

BAF_plock signal can only be deasserted when external BAF PLock is deasserted and correct password is driven to deassert BAF_plock(low_lock[0]) of LOCK0_PGx

84.3.1.9 Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register will become writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits will not come into effect before the life cycle is matured to “OEM Production” or older.

Figure 2112. LOCK1_PG*n* register

0x0104 LOCK1_PG0																	
Offset																	
0x0114 LOCK1_PG1																	
0x0124 LOCK1_PG2																	
0x0134 LOCK1_PG3																	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																	
Reset		x	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		HIGHLOCK															
W																	
Reset		x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾	x ⁽¹⁾

1. If no DCF records have been written to change the lock state, the bit will reset to '1'.

LOCK1_PG*n* register functions are as shown in [Table 1965](#).

Table 1965. LOCK1_PGn register field descriptions

Field	Description
0–15	Reserved
16–31 HIGHLOCK	High Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTest Flash block and copied over during reset. The default value before any DCF record is written, is '1' which indicates a block is locked.

During the life cycle 'Customer Delivery', the user must program a valid password or a DCF record that write this register to unprotect the UTEST for each password group to prevent permanent and irreversible locking of flash blocks.

Warning: Failure to program valid passwords or DCF records for all password groups before the life cycle is matured to "OEM Production" will render the part unusable. However, additional DCF Records can be written to UTest to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible.

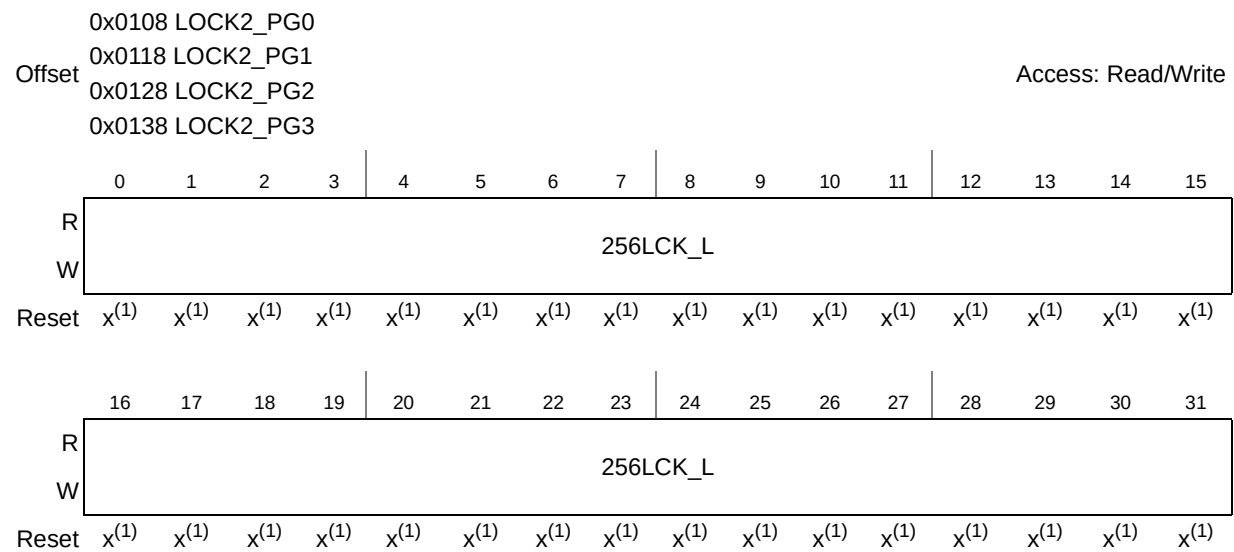
However unless the life cycle is older than "OEM Production", the Flash blocks will always be unlocked, regardless of the values in the password groups.

84.3.1.10 Password Group *n* - Lock 2 status register (LOCK2_PGn)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register will become writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits will not come into effect before the life cycle is matured to "OEM Production" or older.

Figure 2113. LOCK2_PGn register



1. If no DCF records have been written to change the lock state, the bit will reset to '1'.

Note: Lock registers can only be accessed with 32-bit operations.
LOCK2_PGn register functions are as shown in Table 1966.

Table 1966. LOCK2_PGn register field descriptions

Field	Description
0–31 256LCK_L	Lower 256KByte Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

The initial value of the lock bits is set via DCF records stored in UTest Flash block and copied over during reset. The default value before any DCF record is written, is '1' which indicates a block is locked.

During the life cycle 'Customer Delivery', the user must program a valid password or a DCF record that write this register to unprotect the UTest for each password group to prevent permanent and irreversible locking of flash blocks.

Warning:

Failure to program valid passwords or DCF records for all password groups before the life cycle is matured to “OEM Production” will render the part unusable. However, additional DCF Records can be written to UTest to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible.

However unless the life cycle is older than “OEM Production”, the Flash blocks will always be unlocked, regardless of the values in the password groups.

84.3.1.11 Password Group n - Lock 3 status register (LOCK3_PG n)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the PGL bit will be cleared and the Lock registers will become writable.

The LOCK bits will not come into effect before the life cycle is matured to “OEM Production” or older.

Figure 2114. LOCK3_PG n Register

0x010C LOCK3_PG0																
Offset	0x011C LOCK3_PG1															
	0x012C LOCK3_PG2															
	0x013C LOCK3_PG3															
	Access: Read/Write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	RL4 (1)	RL3 (1)	RL2 (1)	RL1 (1)	RL0 (1)
W																
Reset	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	0	1	1	1	1	0	0	0	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256LCK_U															
W																
Reset	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽²⁾	x ⁽³⁾

1. Not all devices include this field. See the device reference manual's chip configuration chapter for details.
2. Reset value depends on life cycle. If life cycle is Customer delivery or earlier, the value will be 0, otherwise 1.
3. If no DCF records have been written to change the lock state, the bit will reset to '1'.

LOCK3_PG n register functions are as shown in [Table 1967](#).

Table 1967. LOCK3_PGn register field descriptions

Field	Description
0 PGL	Password Group Lock This bit may be set in software to lock the password group, when the device is older than Customer Delivery. Before that the value of the bit is always 0. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally. (If life cycle is Customer Delivery or earlier, this bit has no effect and the registers are always unlocked.)
1 DBL	Debug Interface Lock 0 Debug interface is unlocked. Accessed to all JTAG clients is allowed (Except HSM) 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked
2 MO	Master Only 0 All masters can modify the passgroup settings if PGL is cleared 1 Only the Master which unlocked the passgroup can change the passgroup settings
3	Reserved
4-7 MSTR	Master Access Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.
8-10	Reserved
11 RL4	Read Lock Region 4 lock bit. Note that not all devices include this field. Not all devices include this field. See the device reference manual's chip configuration chapter for details. 0Region 4 is not protected 1Region 4 is protected
12 RL3	Read Lock Region 3 lock bit. Note that not all devices include this field. Not all devices include this field. See the device reference manual's chip configuration chapter for details. 0Region 3 is not protected 1Region 3 is protected
13 RL2	Read Lock Region 2 lock bit. Note that not all devices include this field. Not all devices include this field. See the device reference manual's chip configuration chapter for details. 0Region 2 is not protected 1Region 2 is protected
14 RL1	Read Lock Region 1 lock bit. Note that not all devices include this field. Not all devices include this field. See the device reference manual's chip configuration chapter for details. 0Region 1 is not protected 1Region 1 is protected
15 RL0	Read Lock Region 0 lock bit. Note that not all devices include this field. Not all devices include this field. See the device reference manual's chip configuration chapter for details. 0Region 0 is not protected 1Region 0 is protected
16-31 256LCK_U	Upper 256 KB Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: *Lock registers can only be accessed with 32-bit operations.*

The initial value of the lock bits is set via DCF records stored in UTest Flash block and copied over during reset. The default value before any DCF record is written, is '1' which indicates a block is locked.

During the life cycle "Customer Delivery", the user must program a valid password or a DCF record that write this register to unprotect the UTest for each password group to prevent permanent and irreversible locking of flash blocks.

Warning: **Failure to program valid passwords or DCF records for all password groups before the life cycle is matured to "OEM Production" will render the part unusable. However, additional DCF Records can be written to UTest to change the initial value of this register.**

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible.

However unless the life cycle is older than "OEM Production", the Flash blocks will always be unlocked, regardless of the values in the password groups.

Regions will be read protected when write protection and censorship applies and the system is in debug mode. To access the Flash regions, the right passwords need to be provided and the region locks must be cleared.

Once a censored device enters lifecycle "Failure Analysis" read protection is active regardless whether the system is in debug mode or not. Its still possible to unlock Flash regions with the correct passwords, however.

84.4 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

84.4.1 Life cycle and security

The security of the device depends on security state and life cycle of the device as described in Table 1968.

Table 1968. Security features mapped to life cycle

Life Cycle	UTest OTP	Passwords Readable	Crypto Keys Readable	Test Mode Access	HSM available	Debug Access
Customer Delivery	yes	yes	no	no	no	yes
OEM Production	yes	no	no	no	depends on settings	depends on settings

Table 1968. Security features mapped to life cycle (continued)

Life Cycle	UTest OTP	Passwords Readable	Crypto Keys Readable	Test Mode Access	HSM available	Debug Access
In the Field	yes	no	no	no	depends on settings	depends on settings
Failure Analysis	yes	no	no	yes	depends on settings	depends on settings

After the life cycle “OEM Production” debugging access can be prevented by censoring the device. (See Flash documentation.).

If the device is censored and HSM is enabled, then debug access can be granted via HSM challenge/response mechanism—see HSM documentation. If the HSM is not enabled on a censored device, then access is possible via JTAG password.

84.5 Initialization/application information

84.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (see [Section 84.3.1, Register descriptions](#)). However many bits will receive their values based on DCF records stored in the UTest flash memory block.

84.5.2 Modifying life cycle (maturing the device)

To modify the life cycle, the DCF mechanism in the UTest Flash needs to be activated if that hasn't been done already. For this the DCF start record has to be written at location 0x400200.

Following that a life cycle record can be appended at the next free location. If only the start record has been written, that location is at 0x400200 + 0x04, if N records have been written already, the location is at 0x400200 + $(N+1)*0x04$.

The value to be written is determined by [Table 1969](#).

Table 1969. Possible life cycle records

Life Cycle	DCF Value	
	CS/ADDR	Data
Customer Delivery	0x0010_00A0	0x*****FC
OEM Production	0x0010_00A0	0x*****F8
In the Field	0x0010_00A0	0x*****F0
Failure Analysis	0x0010_00A0	0x*****E0

Only the lower 8 bits of the data field are relevant, the others are not used by the PASS.

[Table 1970](#) shows how the life cycle is advanced to “OEM Production” if no other DCF records were written previously.

Table 1970. Moving life cycle to “OEM Production”

ADDR offset	Record	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00F8	Data Record: OEM Production
0x0C	0x0010_00A0	
0x10	0xFFFF_FFFF	Stop Record
0x14	0xFFFF_FFFF	
...	...	

If now the life cycle needs to be advanced to “In the Field”, the UTest content would be modified as shown in [Table 1971](#)

Table 1971. Moving life cycle to *In the Field*

ADDR offset	Record	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00F8	Data Record: OEM Production
0x0C	0x0010_00A0	
0x10	0x0000_00F0	Data Record: In the Field
0x14	0x0010_00A0	
0x18	0xFFFF_FFFF	Stop Record
0x1C	0xFFFF_FFFF	
...	...	

The PASS module will not accept later DCF records to move the life cycle backwards. It's possible to write the 0x0000_00F0 value first, and afterwards the 0x0000_00F8, however the most mature life cycle will take precedence.

84.5.3 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTest flash, apart from the available space for DCF records. It's possible to set and clear them multiple times.

Table 1972. CS/ADDR Values for Lock Registers

Register	DCF CS/ADDR
PW Group 0	
Lock0	0x0010_0100
Lock1	0x0010_0104

Table 1972. CS/ADDR Values for Lock Registers (continued)

Register	DCF CS/ADDR
Lock2	0x0010_0108
Lock3	0x0010_010C
PW Group 1	
Lock0	0x0010_0110
Lock1	0x0010_0114
Lock2	0x0010_0118
Lock3	0x0010_011C
PW Group 2	
Lock0	0x0010_0120
Lock1	0x0010_0124
Lock2	0x0010_0128
Lock3	0x0010_012C
PW Group3	
Lock0	0x0010_0130
Lock1	0x0010_0134
Lock2	0x0010_0138
Lock3	0x0010_013C

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF other fields to 0, the DCF record would have to be added to [Table 1970](#) as shown in [Table 1973](#).

Table 1973. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00F8	Data Record: OEM Production
0x0C	0x0010_00A0	
0x10	0x0000_00FF	PW Group 1, Lock0 Set MIDLOCK to 0xFF
0x14	0x0010_0110	
0x18	0xFFFF_FFFF	Stop Record
0x1C	0xFFFF_FFFF	
...	...	

84.5.4 Censoring/uncensoring the device

There is no restriction on setting and clearing the censorship status in the UTest Flash, apart from the available space for DCF records. It's possible to censor and uncensor multiple times.

The Client Select / Address for the Censorship record is 0x0010_00B0. In order to uncensor the device the data field of the record need to be written with 0x----55AA (upper 16 bits are not compared). Any other value would censor the device.

On delivery the device will be uncensored.

84.5.5 Unlocking the debug interface via JTAG

To open the debug interface, the JTAG password challenge register needs to be written by the debugger with the correct JTAG password.

84.5.6 Production disable

84.5.6.1 Overview and rationale

Production Disable is an optional feature that, once enabled, allows the MCU to functional completely normally without any constraints, except the Debug interface is disabled. Once the customer has decided that the Debug Interface should be re-enabled, the process to re-enable it, makes the MCU unusable in a production environment by adding clock jitter to the baud rate clocks of several serial communications modules (normally CAN and FlexRay). This jitter means those serial communications modules will not be able to communicate on the network.

As this Production Disable feature renders the MCU unusable in the application, inadvertent activation would be disastrous. Therefore a feature is included that prevents this Jitter activation unless an external hardware pin is asserted.

84.5.6.2 Enablement

The feature is enabled via 2 bits in the Production Disable DCF client in the PASS module. One bit enables the Debug Interface Disable. The other bit enables the Hardware Interlock pin. See [Section 84.3.1.6, Production Disable](#) for details.

These bits can be written by adding a DCF Record to the DCF record list in flash UTEST. This DCF Client is configured to accept only one write. Subsequent DCF records that write to this DCF client are ignored.

If no DCF Record writes to this client, the default state is both features is disabled. If, for a particular application, there is no intention of every using this feature, it would be good practice to add a DCF Record that writes this client to 0x0000_0000. This is not essential but ensures the feature can not be Enabled accidentally.

84.5.6.3 Activation

The BAF code is always the first code executed out of reset. The BAF code as part of the normal boot sequence, copies the 'Clock Jitter Activation' constant value from a location in the BAF code space, to the Clock Jitter Register in the PASS module (See [Section 84.3.1.7, Clock Jitter Enable](#) for details).

This register is write once. The default value for the Clock Jitter Activation constant in BAF flash is 0xFFFF_FFFF (erased state)

To enable the Clock Jitter feature, the value of the Clock Jitter Activation constant in BAF flash should be programmed to make the most significant bit '0'. After the next reset, the BAF will copy this updated value to the Clock Jitter Register and the feature will be activated.

The BAF flash block is assigned as OTP and it is protected from programming by a bit in the PASS LOCK0 registers; but the BAF flash block can further be protected from programming by an external hardware pin. If the hardware interlock feature is enabled, while this pin is logic '0' the BAF flash block can not be programmed.

84.5.6.4 Truth table

[Table 1974](#) shows the relationship between various input conditions and the state of the Debug Interface & Baud Rate Clock.

Table 1974. Truth Table of Production Disable Feature

Production Disable[Debug Disable]	Clock Jitter Activation Constant	JTAG Password, Censorship, LOCK3[DBL]	Debug Interface State	Baud Rate Clock State
0	Don't Care		Dependent on JTAG Password or Censorship or LOCK3[DBL]	Normal
1	0b1xxxxxx_(24 more x)	Don't Care	Disabled	Normal
1	0b0xxxxxx_(24 more x)		Dependent on JTAG Password or Censorship or LOCK3[DBL]	Jittering

85 Tamper Detection Module (TDM)

85.1 Overview

Note: Unless otherwise noted, any information related to the flash memory makes reference to the FMC flash memory.

The Tamper Detection Module provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. The mechanism does not *prevent* an erase operation on any block within a TDR—it requires a record to be written before the operation can execute. Collectively, the records are referred to as a “diary”. At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 6 TDRs can be defined via DCF records.

[Figure 2115](#) shows the block diagram of the TDM and its interface to other system components.

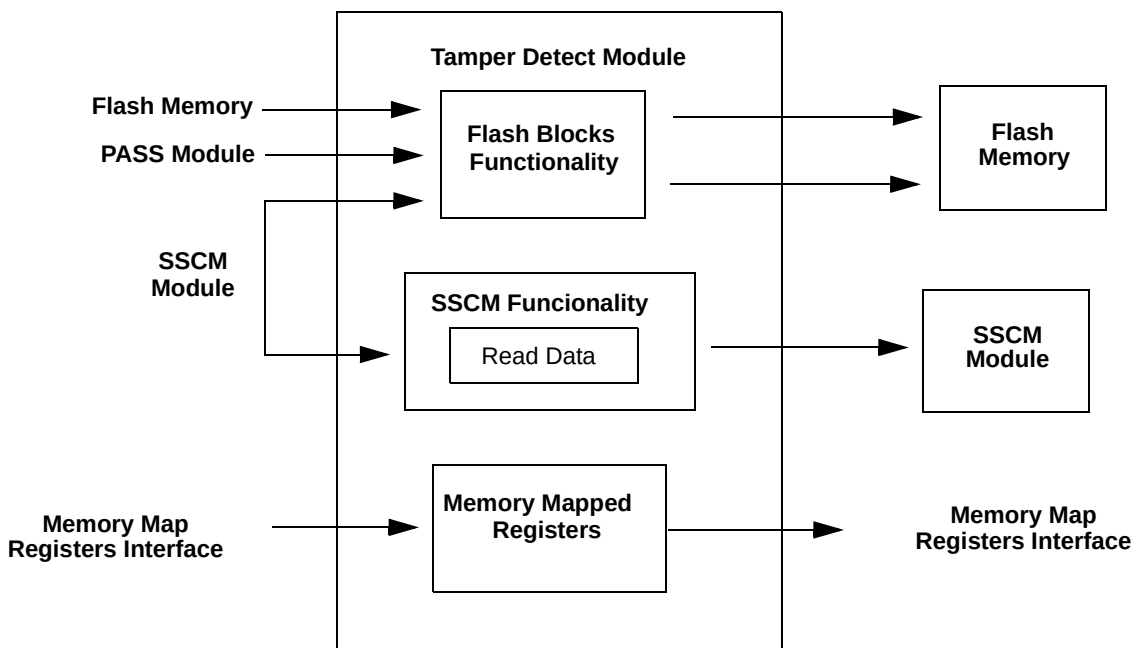


Figure 2115. Tamper Detection Module block diagram

Security information is passed to the TDM by the SSCM module. Among the data are the OTP (One Time Programmable) settings for each of the flash Code sectors, the base address for the diary in the flash and the block assignments for the TDRs. The TDM module in turn provides three dedicated signals:

- Two 128-bit buses to the flash memory
- One 32-bits bus to the SSCM module.

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are

already erased can be programmed. Over-programming is not possible. The PASS module provides the control for erase operation for each flash block as well as for each region of the flash memory, and the PASS settings have priority over the settings of the TDM.

The flash block indicates to the TDM the address of the programmed word. This address is internally decoded and provides a mean to verify whether the block is part of any TDRs. The memory map registers interface provides a mean to the user to read information of the TDM registers.

85.2 Features

The Tamper Detection Module supports these distinctive features:

- Erase Counter (diary) per TDR to record up to 256 write attempts on the flash blocks;
 - The diary is divided into 6 parts (TDRs), each consisting of 256 64-bit records.
 - The diary block should be assigned as OTP region in flash.
 - The diary area can store any type of relevant information.
- Tamper Detection Region Block Assignment
 - Any flash block can be assigned to any TDR and be controlled.
- 32-bit Erase Lock registers for each TDR to the flash memory.
- 32-bit OTP Enable registers to disable the erase operation for an entire flash block
- Override TDR via DCF records
 - It allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation.
- Memory Map Registers Interface for reading via software of specific registers.

85.3 External signal description

There are no external signals driving or being driven by the TDM.

85.4 Memory map and register description

This section provides a detailed description of all registers accessible in the TDM module memory map.

85.4.1 DCF client

[Table 1975](#) gives an overview on the TDM DCF clients implemented.

Table 1975. DCF client address map

DCF Client Address ADDR[16:2]	DCF Client Name	Description	Width
0x0	Diary Base Address ^{(1),(2)}	Stores the base address of the diary being accessed	32
0x1	Tamper Region Override ⁽³⁾	Stores information of which TDR is to be overridden (unlocked for erase)	6

Table 1975. DCF client address map (continued)

DCF Client Address ADDR[16:2]	DCF Client Name	Description	Width
0x8 - (0x8 + 3)	OTPEN[nb] - OTP Enable ⁽³⁾	Stores information of which flash blocks is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	32
0x14 - (0x14 + (4 * 6) - 1))	TDR[n]_LOCK[m] Region Assignment ⁽³⁾	Stores information of which flash block is assigned to a TDR. - n specifies a TDR - m specifies the flash blocks as per the LOCK registers of the flash memory.	32

1. DCF client is write-once.
2. DCF client bits can be written from 1-> 0 only.
3. DCF client bits can be written from 0-> 1 only.

85.4.2 Memory Mapped Registers

Table 1976 shows the memory map for the memory mapped registers

Table 1976. TDM IPS registers address map

Offset from TDM_BASE_PP	Register	Access	Width	Reset Value
0x0000_0000	TDR Status Register (TDRSR)	Read	6	0x0000_003F
0x0000_0004	Last Flash Programmed Address Register (LFPAR)	Read	24	0x0000_0000

The convention in Figure 2116 serves as a key for the individual register diagrams.

Always Reads 1	1	Always Reads 0	0	R/W bit	BIT	Read-Only bit	BIT	Write-Only bit	BIT	Write 1 to clear	BIT	Self-clear bit	0	N/A	BIT
----------------	---	----------------	---	---------	-----	---------------	-----	----------------	-----	------------------	-----	----------------	---	-----	-----

Figure 2116. Key for register rield

Note: Reserved bits will be read as 0s. Attempts on writing to reserved bits can generate unpredictable results for DCF clients and transfer errors for memory mapped registers.

85.4.3 DCF client and register descriptions

This section describes in address order all the TDM DCF clients and registers. Details of the register bit and field function follow the register diagrams in bit order.

85.4.3.1 DCF clients

85.4.3.1.1 Diary Base Address (DBA) DCF client

This DCF client holds base address information of the diary. [Figure 2117](#) shows the diagram for this register.

Warning: This DCF client is implemented as Write Once and is writable for 1 to 0 only. After one DCF Record has written to this client, all subsequent writes are ignored.

Figure 2117. Diary Base Address (DBA) DCF client

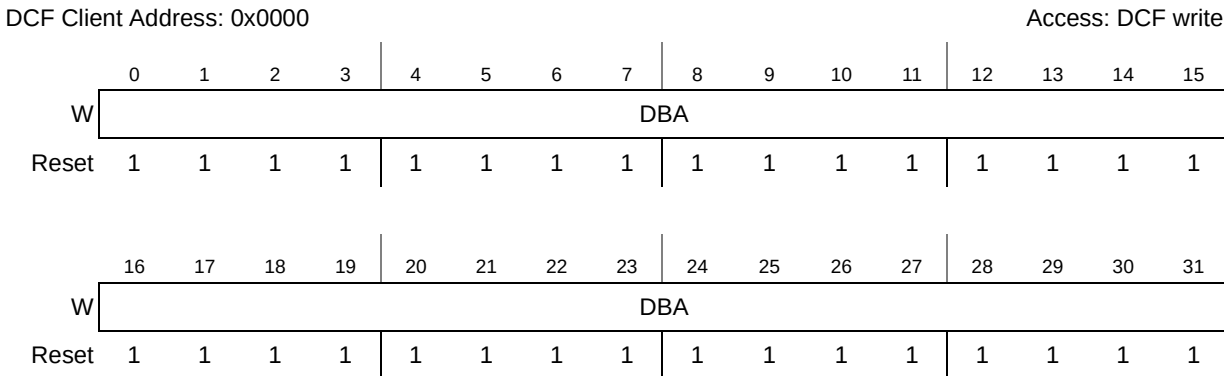


Table 1977. DBA field descriptions

Field	Description
0–31 DBA	Diary Base Address. This field holds the base address of the diary. This information is then used to verify if the diary being programmed matches any TDR region.

85.4.3.1.2 Tamper Region Override (TO) DCF client

This DCF client holds override information for each TDR. [Figure 2118](#) shows the diagram for this register.

Warning: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

Figure 2118. Tamper Region Override (TO) DCF client

DCF Client Address: 0x0001

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1978. TO field descriptions

Field	Description
0–25 Reserved	Reserved
26–31 TOE[5:0]	TDR Override Enable. Each TOE field, where <i>n</i> represents the TDRs, holds information of the Override Enable status for each TDR. 0 Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written. 1 Diary override. The diary for this TDR can be erased WITHOUT writing to the diary.

85.4.3.1.3 One Time Programmable Enable (OTPEN0) DCF client

This DCF Client holds the OTP information for the Low and Mid blocks of the flash memory. This register mirrors the LOCK0 register of the flash memory. [Figure 2119](#) shows the diagram for this register.

Warning: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored
UTEST is always assigned as OTP

Figure 2119. One Time Programmable Enable (OTPEN0) DCF client

DCF Client Address: 0x0008

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	LOWOTPEN[11:0]											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	MIDOTPEN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1979. OTPEN0 field descriptions

Field	Description
0–3 Reserved	
4–15 LOWOTPEN	Low Block OTP Enable. Each bit this field is related to the Low Blocks starting at LOWOTPEN[0] with 16 KB block region, then the 32 KB, and lastly the 64 KB. This continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP
16–31 MIDOTPEN	Mid Block OTP Enable. Each bit this field is related to the Mid Blocks starting at MIDOTPEN[0] with 16 KB block region, then the 32 KB, and lastly the 64 KB. This continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

85.4.3.1.4 One Time Programmable Enable (OTPEN1) DCF client

This DCF client holds the OTP information for the High blocks of the flash memory. This DCF client mirrors the LOCK1 register of the flash memory. [Figure 2120](#) shows the diagram for this DCF Client.

Warning: This DCF Client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

Figure 2120. One Time Programmable Enable (OTPEN1) DCF client

DCF client Address: 0x0009

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	HIGHOTPEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1980. OTPEN1 field descriptions

Field	Description
0–15 Reserved	
16–31 HIGHOTPEN	High Block OTP Enable. Each bit this field is related to the High Blocks starting at HIGHOTPEN[0] with 16 KB block region, then the 32 KB, and lastly the 64 KB. This continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

85.4.3.1.5 One Time Programmable Enable (OTPEN2) DCF client

This DCF client holds the OTP information for the first thirty-two 256 KB blocks of the flash memory. This DCF client mirrors the LOCK2 register of the flash memory. [Figure 2121](#) shows the diagram for this DCF client.

Warning: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

Figure 2121. One Time Programmable Enable (OTPEN2) DCF client

DCF Client Address: 0x000A

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	256KBOTPEN[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	256KBOTPEN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1981. OTPEN2 field descriptions

Field	Description
0–31 256KBOTPEN	256 KB Block OTP Enable. Each bit this field is related to the 256 KB Blocks starting at 256KBOTPEN[0] and continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

85.4.3.1.6 One Time Programmable Enable (OTPEN3) DCF client

This DCF Client holds the OTP information for the last sixteen 256 KB blocks of the flash memory. This DCF Client mirrors the LOCK3 register of the flash memory. [Figure 2122](#) shows the diagram for this DCF Client.

Warning: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

Figure 2122. One Time Programmable Enable (OTPEN3) DCF client

DCF client Address: 0x000B

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	256KBOTPEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1982. OTPEN3 field descriptions

Field	Description
0–15 Reserved	
16–31 256KBOTPEN	256 KB Block OTP Enable. Each bit this field is related to the 256 KB Blocks starting at 256KBOTPEN[0] and continues until all blocks are accounted for. 0Corresponding flash blocks are not assigned as OTP 1Corresponding flash blocks are assigned as OTP

85.4.3.1.7 Tamper Region Assignment DCF client (TDR_n_LOCK0)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF client however holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. Therefore, the diagram of this register mirrors the LOCK0 register in the flash memory. [Figure 2123](#) depicts the TDR_n_LOCK0 DCF client.

Warning: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

Figure 2123. Tamper Region Assignment DCF client (TDRn_LOCK0)DCF client Address: $0x14 + ((n \times 4) + 0)$

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	LOWTDRAGN											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	MIDTDRAGN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1983. TDRn_LOCK0 field descriptions

Field	Description
0–3 Reserved	
4–15 LOWTDRAGN	Low Block Tamper Region Assignment. Each bit this field is related to the Low Blocks starting at LOWTDRAGN[0] with 16 KB block region, then the 32 KB, and lastly the 64 KB. This continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn
16–31 MIDTDRAGN	Mid Block Tamper Region Assignment. Each bit this field is related to the Mid Blocks starting at MIDTDRAGN[0] with 16 KB block region, then the 32 KB, and lastly the 64 KB. This continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

85.4.3.1.8 Tamper Region Assignment DCF client (TDRn_LOCK1)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (n varies from zero to 5). This DCF client however holds the assignment information for only those blocks specified in the LOCK1 register in the flash memory. Therefore, the diagram of this DCF client mirrors the LOCK1 register in the flash memory. [Figure 2124](#) depicts the TDRn_LOCK1 DCF client.

Warning: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

Figure 2124. Tamper Region Assignment DCF client (TDRn_LOCK1)DCF client Address: $0x14 + ((n \times 4) + 1)$

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	HIGHTDRAGN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1984. TDRn_LOCK1 field descriptions

Field	Description
0–15 Reserved	
16–31 HIGHTDRAG N	High Block Tamper Region Assignment. Each bit this field is related to the High Blocks starting at HIGHTDRAGN[0] with 16 KB block region, then the 32 KB, and lastly the 64 KB. This continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

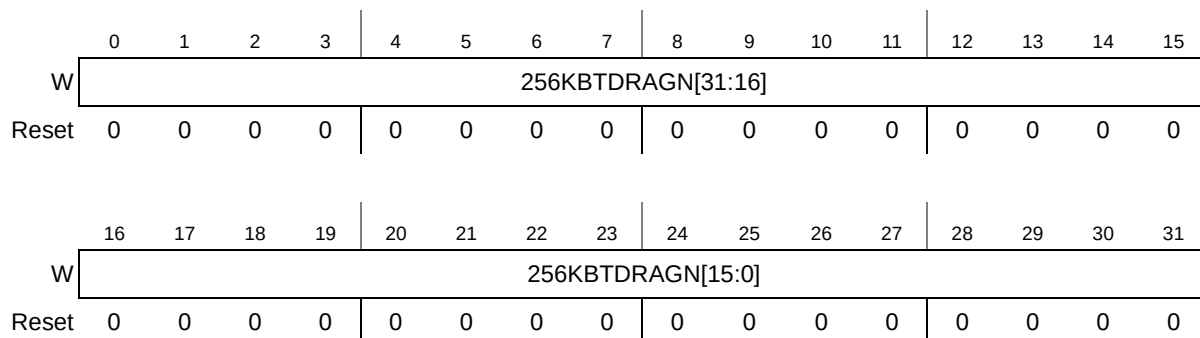
85.4.3.1.9 Tamper Region Assignment DCF client (TDRn_LOCK2)

This DCF Client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (n varies from zero to 5). This DCF Client however holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. Therefore, the diagram of this DCF Client mirrors the LOCK2 register in the flash memory. [Figure 2125](#) depicts the TDRn_LOCK2 DCF Client.

Warning: This DCF Client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

Figure 2125. Tamper Region Assignment DCF client (TDR_n_LOCK2)DCF client Address: $0x14 + ((n \times 4) + 2)$

Access: DCF write

**Table 1985. TDR_n_LOCK2 field descriptions**

Field	Description
0–31 256KBTDRAIN	256 KB Block Tamper Region Assignment. Each bit this field is related to the 256 KB blocks starting at 256 KBTDRAIN[0] and continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned to the TDR _n 1 Corresponding flash blocks are assigned to the TDR _n

85.4.3.1.10 Tamper Region Assignment DCF client (TDR_n_LOCK3)

This DCF Clients holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF Clients however holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. Therefore, the diagram of this DCF Clients mirrors the LOCK3 register in the flash memory. [Figure 2126](#) depicts the TDR_n_LOCK3 DCF Clients.

Warning: This DCF Clients is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

Figure 2126. Tamper Region Assignment DCF client (TDR_n_LOCK3)DCF client Address: $0x14 + ((n \times 4) + 3)$

Access: DCF write

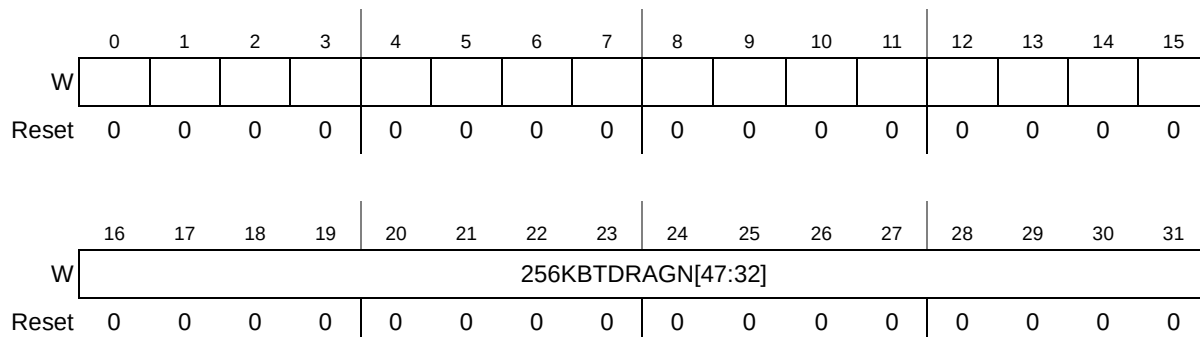


Table 1986. TDRn_LOCK3 field descriptions

Field	Description
0–15 Reserved	
16–31 256KBTDRAGN	256KB Block Tamper Region Assignment. Each bit this field is related to the 256 KB blocks starting at 256KBTDRAGN[0] and continues until all blocks are accounted for. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

85.4.3.2 Memory mapped registers

85.4.3.2.1 TDR Status Register (TDRSR)

This register holds information for the status of each TDR. There is one bit in the register for each TDR. The status bit is a combination of the unlock state and the TDR AND the TDR Override bit. The default state of each status bit is '1', meaning that Erase operation for all flash blocks associated to the TDRs is disabled. The status bit for a TDR will be set to '0' (Indicating that Erase operation for all flash blocks associated to the TDR can be Erased), if either a successful programming operation to the TDR diary region has been performed OR the TDR Override bit for that TDR is set.

TDRSR is a read-only register and write shall have no effect, and attempts on doing so will generate a transfer error.

Figure 2127. TDR Status and Diary Override Register (TDRSR)

Offset: 0x00 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDR SR5	TDR SR4	TDR SR3	TDR SR2	TDR SR1	TDR SR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Table 1987. TDRSR field descriptions

Field	Description
0–25 Reserved	
26–31 TDRSR[5:0]	Each bit of this field represent the status of a TDR in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by either a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

85.4.3.2.2 Last Flash Programmed Address Register (LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Write to this register shall be ignored, but it will generate a transfer error.

Figure 2128. Last Flash Programmed Address Register (LFPAR)

Offset: 0x04

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LFPAR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LFPAR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1988. LFPAR field descriptions

Field	Description
0–7 Reserved	
8–31 LFPAR	Last Flash Programmed Address. These bits represent the address of the last successful programming operation.

85.5 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

85.5.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the Diary is determined by a DCF Record.

The diary is divided into 6 parts where each part consists of 256 x 64-bit records, to record until 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the Diary should reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4 x 32 bits registers. These registers mirror exactly the LOCK registers bits from the flash module.

Each of the m registers is loaded at boot time by the SSCM, which defines if the associated flash block is Locked for Erase in a particular TDR:

- '1' means the flash block is locked for erase
- '0' means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR. As the diary should be OTP, successive writes are allowed on unprogrammed double words only. This ensure that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double word program operation, the flash module indicates to the TDM the address of the programmed double word. TDM determines whether the programmed data is done within a TDR and unlocks the blocks for erase operation if data are programmed a diary. TDM ensures that this erase operation on unlocked blocks is allowed until the flash indicates a program erase complete interrupt or till next hard reset.

Note: All blocks associated to a TDR where an erase operation is allowed can be erased all together.

TDM has no control of how much data a diary has loaded. It is a responsibility for the user to verify if the diary is full. There is also no restriction to what type of data can be programmed in the diary. If diary is detected to be full, TDM provides a mean to disable the tamper detect mechanism so that erase operations can be performed to a TDR without any diary entry. In this case, the user is required to program the Tamper Region Override Register for the corresponding TDR.

The data programmed in the diary location is flexible, and may include count information or other important customer information.

Software has to check whether the diary is full. Once the diary area for a TDR is full, the software has to program a DCF record for overriding blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry

The setting from "override tamper region lock" DCF record (see [Section 85.5.3, Diary](#)) has priority in the settings in the TDRx_LOCK0-3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDRx_LOCK n bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, the settings from PASS_LOCKx_PG n registers have priority on TDRx_LOCKs setting. For example, if a block is password-protected for write operations, the TDR lock setting is ignored.

A schematic representation of the diary is depicted in [Figure 2129](#).

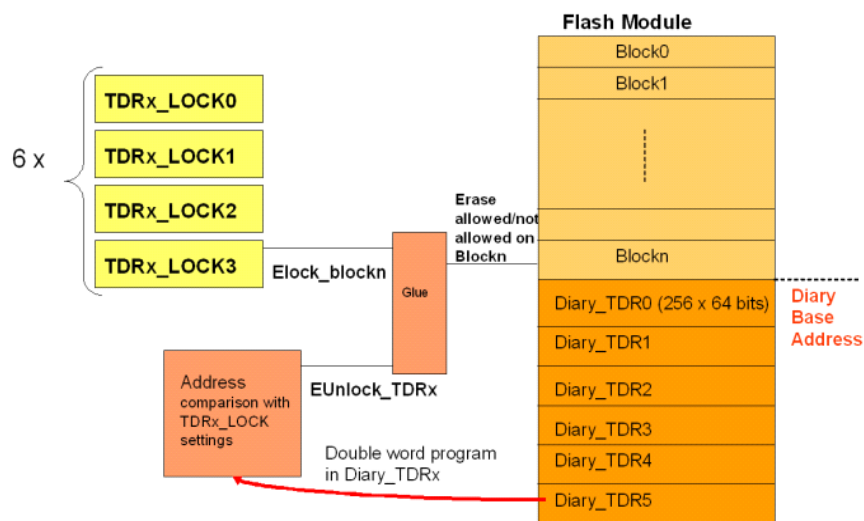


Figure 2129. Diary implementation

Note: The settings in the Tamper Region Override Register have priority over the $TDRn_LOCKm$ Registers. The SSCM module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via TDM, but the same block are programmed to be locked for erase operation in SSCM, the blocks end up being locked for erase operation.

85.5.2 Assigning blocks to Tamper Detect Regions (TDR)

Each TDR is a group of 4 x 32 bit registers. These registers mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as $TDRn_LOCKm$ where n is the number of a tamper region, i.e., 0-5.

In these registers:

- '1' means that the corresponding flash block is assigned to this TDR
- '0' means that the corresponding flash block is not assigned to this TDR (Default)

These DCF clients are only writable once during reset and the bit default state is 0. These registers are not visible to software. They can not be read nor written. They can only be initialized by DCF record within UTest block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that write these registers.

Note: The DCF clients are used to load the data recorded in the UTest block in device configuration format, loaded previously by the SSCM module, into registers. Consult the DCF Client documentations for further details.

85.5.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 2 KB, and the minimum size of any programming operation is 8 bytes. Therefore each diary can hold a maximum of 256 64-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core (also from HSM, if present).

Note: To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once during reset and the default bit state is 1.

The DCF clients can not be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

85.5.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records is defined by the customer. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes. Therefore the diary can hold a maximum of 256 x 8 byte entries.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once during reset and the default bit state is '0'.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 2 KB section of the diary. With 6 TDRs, 12 KB of diary space is required.

The diary base address must be at a 4 KB boundary. Therefore the least significant 12 bits of the base address must all be '0' (Hence only 20 bit DCF client is required).

The Base address of the diary is represented by:-

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xxxx_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

85.5.5 Diary Base Address Verification

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. See [Section 85.4.3.2.1, TDR Status Register \(TDRSR\)](#) for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programmed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning "TDR Erase Enabled".

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 6 status bits can be read through the TDRSR. See [Section 85.4.3.2.1, "TDR Status Register \(TDRSR\)"](#). The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. See [Section 85.4.3.2.2, "Last Flash Programmed Address Register \(LFPAR\)"](#).

85.5.5.0.1 Diary management

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location that can be written. After a maximum of 256 records are programmed to the diary, a TDR diary is full. Full means that at least 1 bit has been programmed to '0' in each 64-bit double word within the 2 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits - one for each TDR. See [Table 121](#) for TDM DCF client details.

The Diary Override bits are by default set to '0', which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to '1'. Writes to '0' are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to '1'. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

85.5.6 One Time Programmable (OTP)

85.5.6.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and that only 64-bit double words that are already erased (i.e., flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block will become OTP after the next reset.

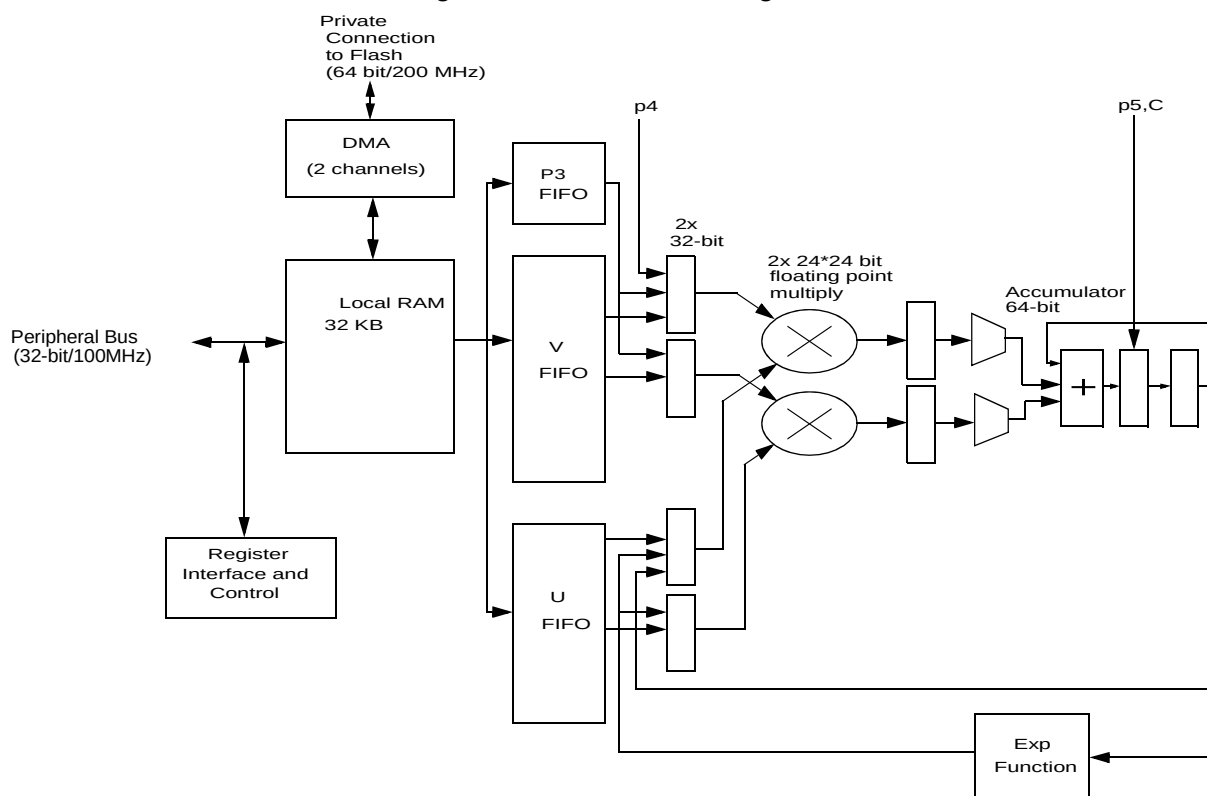
86 Arithmetic Modeling Unit (AMU)

86.1 Introduction

The Arithmetic Modeling Unit (AMU) is a customer arithmetic calculation unit. It is used for designing complex multi-dimensional data models. The AMU incorporates a variant of neural network technology to accomplish calculation tasks. The AMU contains two separate channels, which are timing-independent and register-independent. They both share the same arithmetic logic. Operations are interleaved on a data path containing two multipliers, one accumulator, and one exponential function, besides a number of blocks to feed these arithmetic blocks with data. The interleaving of the channels is such that operations on one channel do not influence the timing on the other channel.

A block diagram of the AMU block is given in [Figure 2130](#). The block consists of a local 32 KB RAM with 64-bit ECC, a hard-wired data path to calculate the AMU function, and DMA channels to do data transfers from flash to the AMU local RAM. The AMU local RAM is also visible over the peripheral bus to the system. The data path is superscalar, in that it contains two multipliers. This allows two multiplications to be done in one cycle for one channel.

Figure 2130. AMU block diagram



86.1.1 Features

- Two independent calculation channels. The two independent channels allow two concurrent processes to use the AMU to do a calculation, with a guarantee in hardware there is no influence between the two processes.
- Two independent DMA channels. The DMA channels are high-bandwidth direct memory access channels, that allow transfer of flash data to local RAM.

86.2 Memory map and register definition

The AMU module provides an IPS programming model mapped to standard 16 KB on-platform peripheral slots. The local AMU RAM is accessible for read and write via the IPS interface and is mapped to four contiguous and properly aligned 16 KB peripheral slots for a maximum RAM size of 64 KB. The programming model can only be referenced in supervisor mode using 32-bit (word) accesses. Attempted references while in user mode, using different access sizes, or to undefined (reserved) addresses generate an IPS error termination. All programming model register reads and writes complete in a single IPS clock cycle.

86.2.1 AMU memory map

The following table shows the memory map for the AMU module.

Table 1989. AMU memory map

Address Offset	Register	Access	Reset Value	Section/Page
0x00000 – 0x0FFFF	Direct access to Local RAM	RW	0x0000_0000	
0x10000	VCH0—AMU V Pointer Channel 0 register	RW	0x0000_0000	86.2.2.1/3651
0x10004	UCH0—AMU U Pointer Channel 1 register	RW	0x0000_0000	86.2.2.2/3652
0x10008	P1CH0—AMU P1 Pointer Channel 0 register	RW	0x0000_0000	86.2.2.2/3652
0x1000C	P2CH0—AMU P2 Pointer Channel 0 register	RW	0x0000_0000	86.2.2.2/3652
0x10010	P3CH0—AMU P3 Pointer Channel 0 register	RW	0x0000_0000	86.2.2.5/3654
0x10014	P8CH0—AMU P8 Channel 0 register	RW	0x0000_0000	86.2.2.6/3655
0x10018	P6CH0—AMU P6 Channel 0 register	RW	0x0000_0000	86.2.2.7/3655
0x1001C	NCH0—AMU Ncrossing Channel 0 register	RW	0x0000_0000	86.2.2.8/3656
0x10020	P7CH0—AMU P7 Channel 0 register	RW	0x0000_0000	86.2.2.9/3657
0x10024	P4CH0—AMU P4 Channel 0 register	RW	0x0000_0000	86.2.2.10/3657
0x10028	P5CH0—AMU P5 Channel 0 register	RW	0x0000_0000	86.2.2.10/3657
0x1002C	RESCH0—AMU Result Channel 0 register	RW	0x0000_0000	86.2.2.11/3658
0x10030	V32_0 ⁽¹⁾ —AMU 32-bit V vector Channel 0 register	RW	0x0000_0000	86.2.2.12/3659
0x10034	V64L_0—AMU 64-bit V Vector Least Significant Bytes Channel 0 register	RW	0x0000_0000	86.2.2.13/3660

Table 1989. AMU memory map (continued)

Address Offset	Register	Access	Reset Value	Section/Page
0x10038	V64M_0—AMU 64-bit V Vector Most Significant Bytes Channel 0 register	RW	0x0000_0000	86.2.2.13/3660
0x1003C	DSACH0—AMU DMA Source Address Channel 0 register	RW	0x0000_0000	86.2.2.15/3661
0x10040	DDACH0—AMU DMA Destination Address Channel 0 register	RW	0x0000_0000	86.2.2.16/3661
0x10044	DCNTCH0—AMU DMA Transfer Count Channel 0 register	RW	0x0000_0000	86.2.2.17/3662
0x10048	DCTLCH0—AMU DMA Transfer Control Channel 0 register	RW	0x0000_0000	86.2.2.18/3663
0x1004C	STCFG0—AMU Status and Configuration register	RW	0x0000_0000	86.2.2.19/3665
0x10050	ESR0—AMU Error Status Register Channel 0	R	0x0000_0000	86.2.2.20/3668
0x10054	ESR0NC—AMU Error Status Register Channel 0, No Clear on Read	R	0x0000_0000	86.2.2.20/3668
0x10058	DVHCH0—AMU DMA V Vector Holdoff Count Channel 0 register	R	0x0000_0000	86.2.2.21/3669
0x10080	VCH1—AMU V Pointer Channel 1 register	RW	0x0000_0000	86.2.2.1/3651
0x10084	UCH1—AMU U Pointer Channel 1 register	RW	0x0000_0000	86.2.2.2/3652
0x10088	P1CH1—AMU P1 Pointer Channel 1 register	RW	0x0000_0000	86.2.2.2/3652
0x1008C	P2CH1—AMU P2 Pointer Channel 1 register	RW	0x0000_0000	86.2.2.2/3652
0x10090	P3CH1—AMU P3 Pointer Channel 1 register	RW	0x0000_0000	86.2.2.5/3654
0x10094	P8CH1—AMU P8 Channel 1 register	RW	0x0000_0000	86.2.2.6/3655
0x10098	P6CH1—AMU P6 Channel 1 register	RW	0x0000_0000	86.2.2.7/3655
0x1009C	NCH1—AMU Ncrossing Channel 1 register	RW	0x0000_0000	86.2.2.8/3656
0x100A0	P7CH1—AMU P7 Channel 1 register	RW	0x0000_0000	86.2.2.9/3657
0x100A4	P4CH1—AMU P4 Channel 1 register	RW	0x0000_0000	86.2.2.10/3657
0x100A8	P5CH1—AMU P5 Channel 1 register	RW	0x0000_0000	86.2.2.10/3657
0x100AC	RESCH1—AMU Result Channel 1 register	RW	0x0000_0000	86.2.2.11/3658
0x100B0	V32_1—32-bit V Vector Channel 1 register	RW	0x0000_0000	86.2.2.12/3659
0x100B4	V64L_1—64-bit V Vector Least Significant Bytes Channel 1 register	RW	0x0000_0000	86.2.2.13/3660
0x100B8	V64M_1—64-bit V Vector Most Significant Bytes Channel 1 register	RW	0x0000_0000	86.2.2.13/3660
0x100BC	DSACH1—AMU DMA Source Address Channel 1 register	RW	0x0000_0000	86.2.2.15/3661
0x100C0	DDACH1—AMU DMA Destination Address Channel 1 register	RW	0x0000_0000	86.2.2.16/3661

Table 1989. AMU memory map (continued)

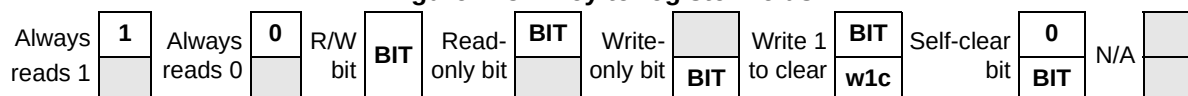
Address Offset	Register	Access	Reset Value	Section/Page
0x100C4	DCNTCH1—AMU DMA Transfer Count Channel 1 register	RW	0x0000_0000	86.2.2.17/3662
0x100C8	DCTLCH1—AMU DMA Transfer Control Channel 1 register	RW	0x0000_0000	86.2.2.18/3663
0x100CC	STCFG1—AMU Status and Configuration register	RW	0x0000_0000	86.2.2.19/3665
0x100D0	ESR1—AMU Error Status Register Channel 1	R	0x0000_0000	86.2.2.20/3668
0x100D4	ESR1NC—AMU Error Status Register Channel 1, No Clear on Read	R	0x0000_0000	86.2.2.20/3668
0x100D8	DVHCH1—AMU DMA V Vector Holdoff Count Channel 1 register	R	0x0000_0000	86.2.2.21/3669

1. p9, as defined in the RfQ, does not have a separate register. Its function is handled by v32_n.

86.2.2 Register Descriptions

This section consists of register descriptions in ascending order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Figure 2131. Key to register fields



86.2.2.1 AMU V Pointer Channel *n* register (VCH_{*n*})

Figure 2132. AMU V Pointer Channel *n* register (VCH_{*n*})

0x10000 VCH0 - V Pointer Channel 0 register

0x10080 VCH1 - V Pointer Channel 1 register

Access: Read/write

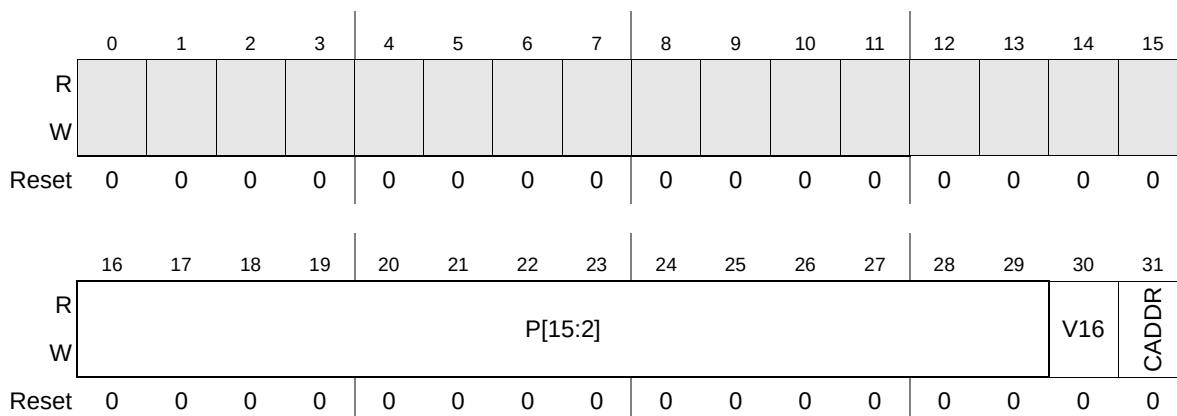


Table 1990. VCH n field description

Field	Description
0:15	Reserved.
16:29 P[15:2]	Pointer[15:0] is the byte address in the local RAM. A 32-bit alignment is forced, as the two LSBs are zero.
30 V16	0: V-vector is 32-bit IEEE floating-point format 1: V-vector is 16-bit IEEE floating-point format
31 CADDR	0: Normal addressing mode 1: Circular address mode. Addressing modulo 256 bytes

86.2.2.2 AMU U Pointer Channel n register (UCH n)

Figure 2133. AMU U Pointer Channel n register (UCH n)

0x10004 UCH0 - U Pointer Channel 0 register

Access: Read/write

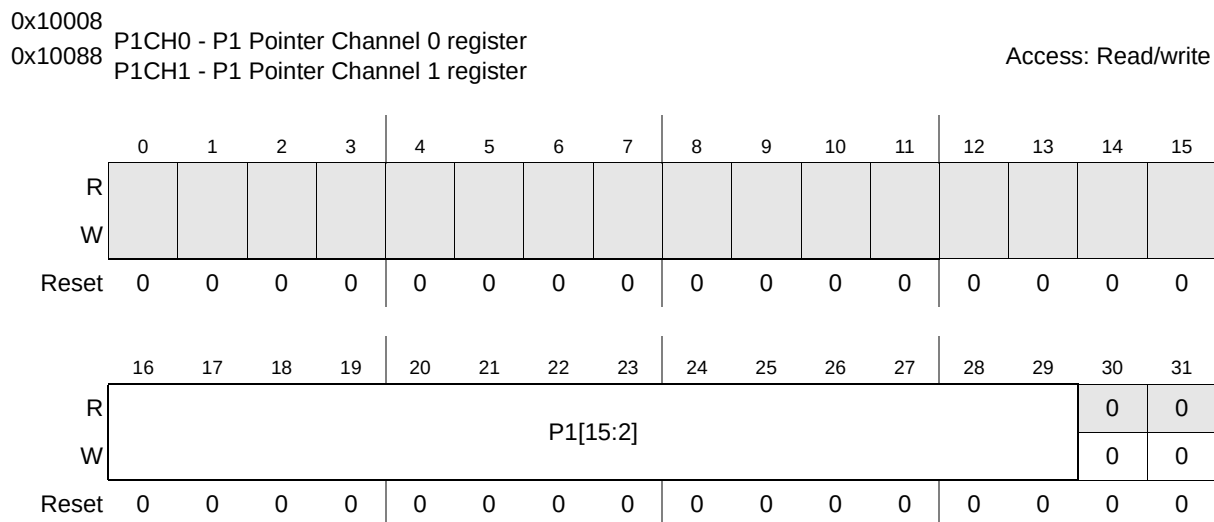
0x10084 UCH1 - U Pointer Channel 1 register

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	P[15:2]														0	0
W															0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1991. UCH n field description

Field	Description
0:15	Reserved.
16:29 P[15:2]	Pointer[15:0] is the byte address in the local RAM. A 32-bit alignment is forced, as the two LSBs are zero.

86.2.2.3 AMU P1 Pointer Channel *n* register (P1CH*n*)Figure 2134. P1 Pointer Channel *n* register (P1CH*n*)Table 1992. P1CH*n* field description

Field	Description
0:15	Reserved.
16:29 P1[15:2]	Pointer to local RAM for p1 vector. P1[15:0] is the byte address. As two LSBs are fixed at zero, an alignment on a 32-bit (word) boundary is enforced.

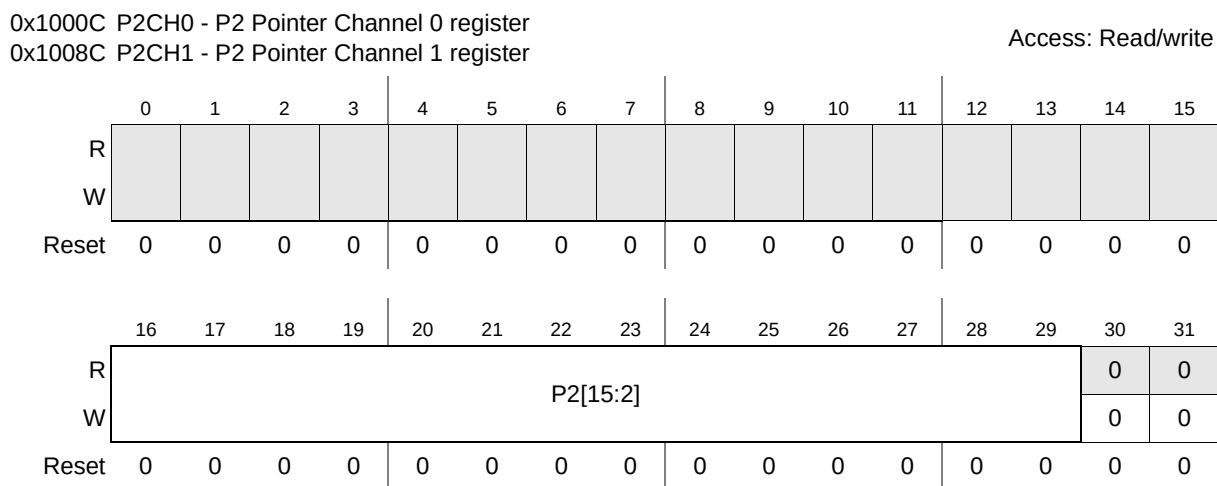
86.2.2.4 AMU P2 Pointer Channel *n* register (P2CH*n*)Figure 2135. AMU P2 Pointer Channel *n* register (P2CH*n*)

Table 1993. P2CH n field description

Field	Description
0:15	Reserved.
16:29 P2[15:2]	Pointer to local RAM for p2 vector. P2[15:0] is the byte address. As two LSBs are fixed at zero, an alignment on a 32-bit (word) boundary is enforced.

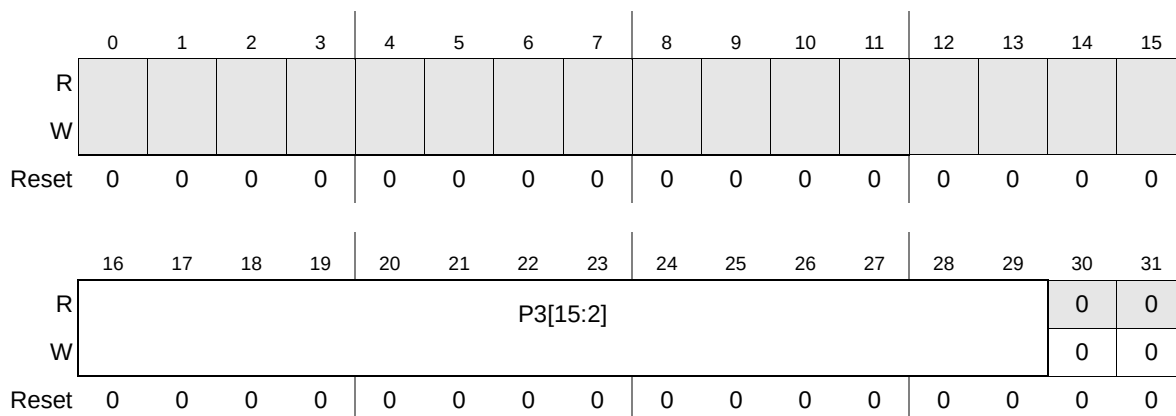
86.2.2.5 AMU P3 Pointer Channel n register (P3CH n)

Figure 2136. AMU P3 Pointer Channel n register (P3CH n)

0x10010 P3CH0 - P3 Pointer Channel 0 register

Access: Read/write

0x10090 P3CH1 - P3 Pointer Channel 1 register

Table 1994. P3CH n field description

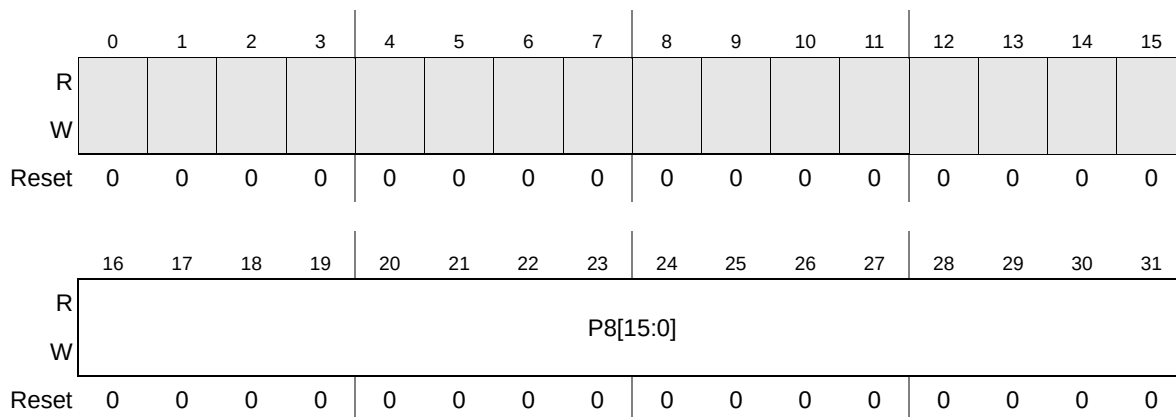
Field	Description
0:15	Reserved
16:29 P3[15:2]	Pointer to local RAM for p3 vector. P3[15:0] is the byte address. As two LSBs are fixed at zero, an alignment on a 32-bit (word) boundary is enforced.

86.2.2.6 AMU P8 Channel *n* register (P8CH*n*)Figure 2137. P8 Channel *n* register (P8CH*n*)

0x10014 P8CH0 - P8 Channel 0 register

0x10094 P8CH1 - P8 Channel 1 register

Access: Read/write

Table 1995. P8CH*n* field description

Field	Description
0:15	Reserved.
16:31 P8[15:0]	Outer loop counter (unsigned integer). This field must be initialized before the start of each calculation with the appropriate value. This field is incremented during the V calculation after each iteration of the outer loop is completed.

86.2.2.7 AMU P6 Channel *n* register (P6CH*n*)Figure 2138. AMU P6 Channel *n* register (P6CH*n*)

0x10018 P6CH0 - P6 Channel 0 register

0x10098 P6CH1 - P6 Channel 1 register

Access: Read/write

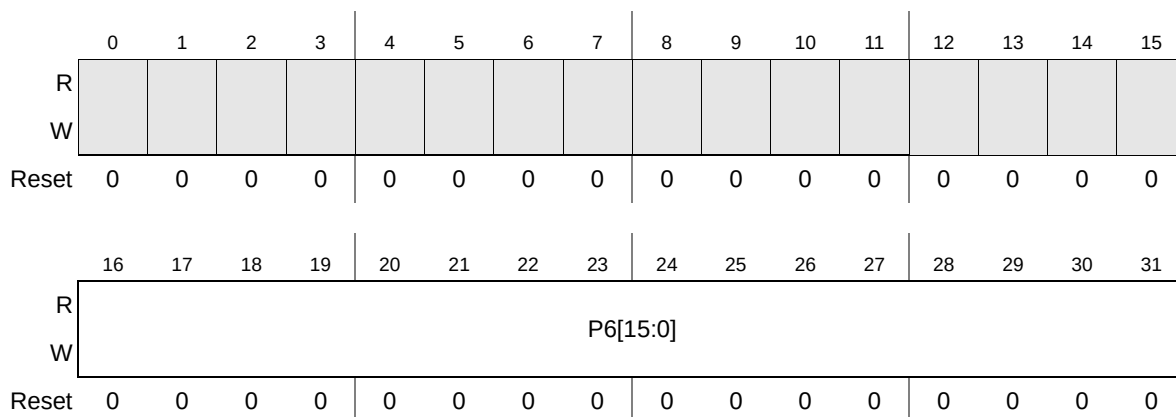


Table 1996. P6CH n field description

Field	Description
0:15	Reserved.
16:31 P6[15:0]	Outer loop count limit (unsigned integer). This field remains static once written.

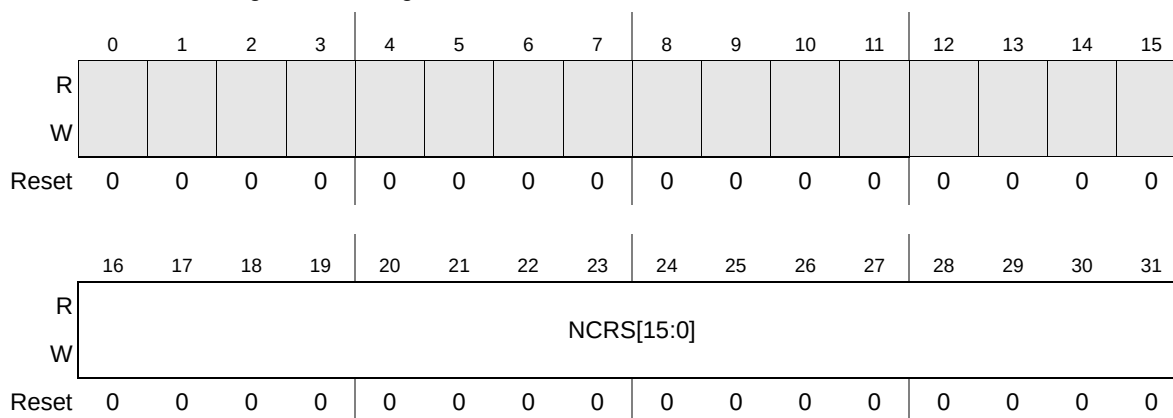
86.2.2.8 AMU Ncrossing Channel n register (NCH n)

Figure 2139. AMU Ncrossing Channel n register (NCH n)

0x1001C NCH0 Ncrossing channel 0 register

Access: Read/write

0x1009C NCH1 Ncrossing channel 1 register

Table 1997. NCH n field description

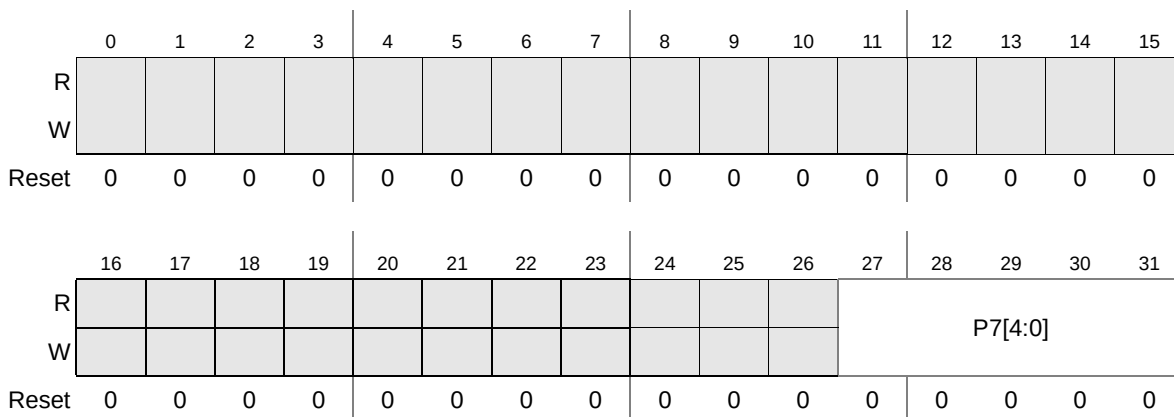
Field	Description
0:15	Reserved.
16:31 NCRS[15:0]	Ncrossing value (unsigned integer). When properly enabled in STCFG n [VSN], the V calculation is stopped after NCRS iterations of the outer loop. After the calculation is stopped, the p8 outer loop count matches the NCRS value set in this register. The intermediate value contained in the V32 register is consistent with this p8/NCRS value. Additionally, when the STCFG n [NIE] control bit is set, an interrupt is triggered when the outer loop count reaches the specified NCRS value. The interrupt enable/disable control is independent of the STCFG n [VSN] setting.

86.2.2.9 AMU P7 Channel *n* register (P7CH*n*)Figure 2140. AMU P7 Channel *n* register (P7CH*n*)

0x10020 P7CH0 - P7 Channel 0 register

0x100A0 P7CH1 - P7 Channel 1 register

Access: Read/write

Table 1998. P7CH*n* field description

Field	Description
0:26	Reserved.
27:31 P7[4:0]	Inner loop count limit (unsigned integer). This field remains static once written.

86.2.2.10 AMU P4 Channel *n* register (P4CH*n*)
AMU P5 Channel *n* register (P5CH*n*)Figure 2141. AMU P4, P5 Channel *n* register

0x10024 P4CH0 - P4 Channel 0 register

0x10028 P5CH0 - P5 Channel 0 register

0x100A4 P4CH1 - P4 Channel 1 register

0x100A8 P5CH1 - P5 Channel 1 register

Access: Read/write

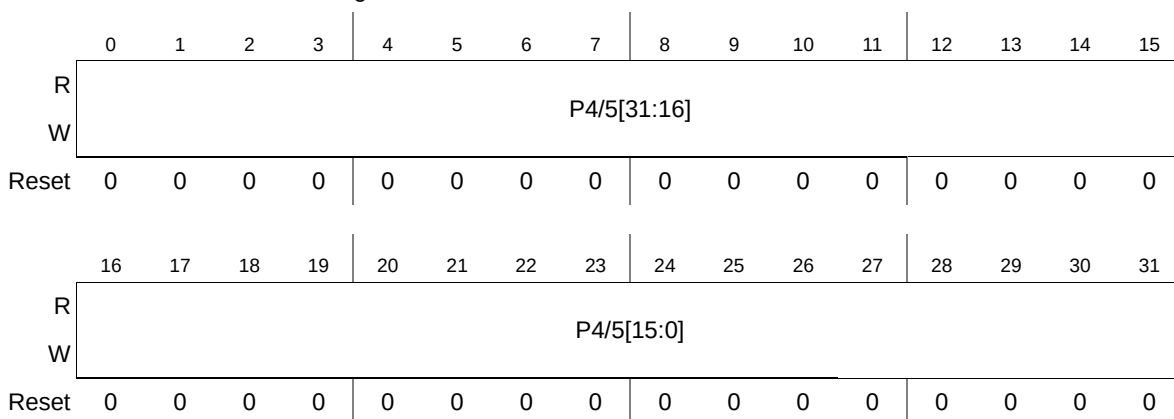


Table 1999. P4CHn, P5CHn field description

Field	Description
0:31 P4[31:0] P5[31:0]	Parameter P4, P5 values, IEEE 32-bit floating point format. These values remain static once written.

86.2.2.11 AMU Result Channel n register (RESCHn)

Figure 2142. AMU Result Channel n register (RESCHn)

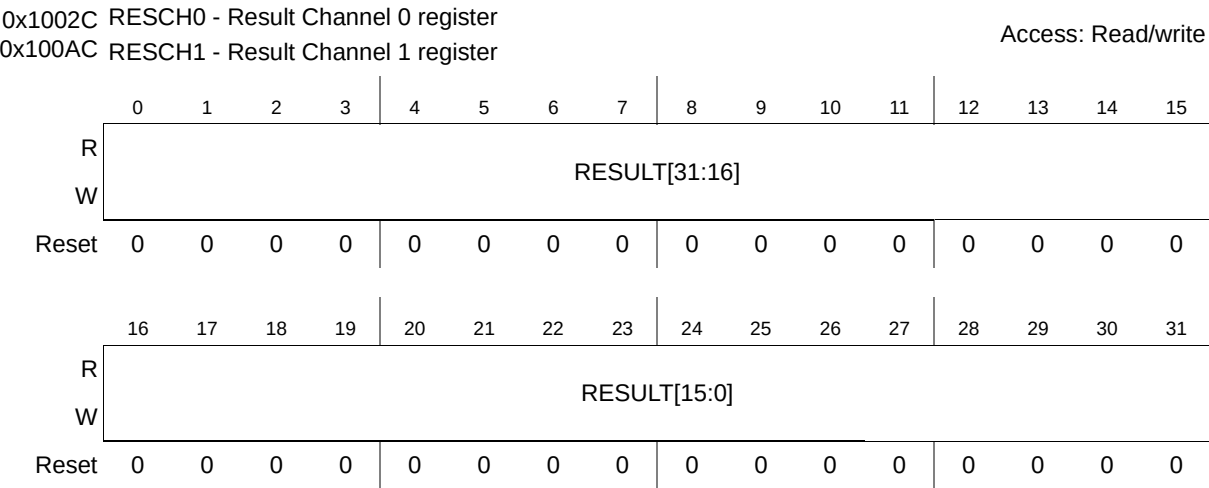


Table 2000. RESCHn field description

Field	Description
0:31 RESULT[31:0]	Result value - IEEE 32-bit floating point notation



86.2.2.12 AMU 32-bit V vector Channel n register (V32_n)

Note: The initial V32 (also known as parameter p9) value must be written before any new calculation is started. The value is not retained, as V32 is updated after each iteration of the outer loop of the calculation.

Figure 2143. 32-bit V Vector Channel n register (V32_n)

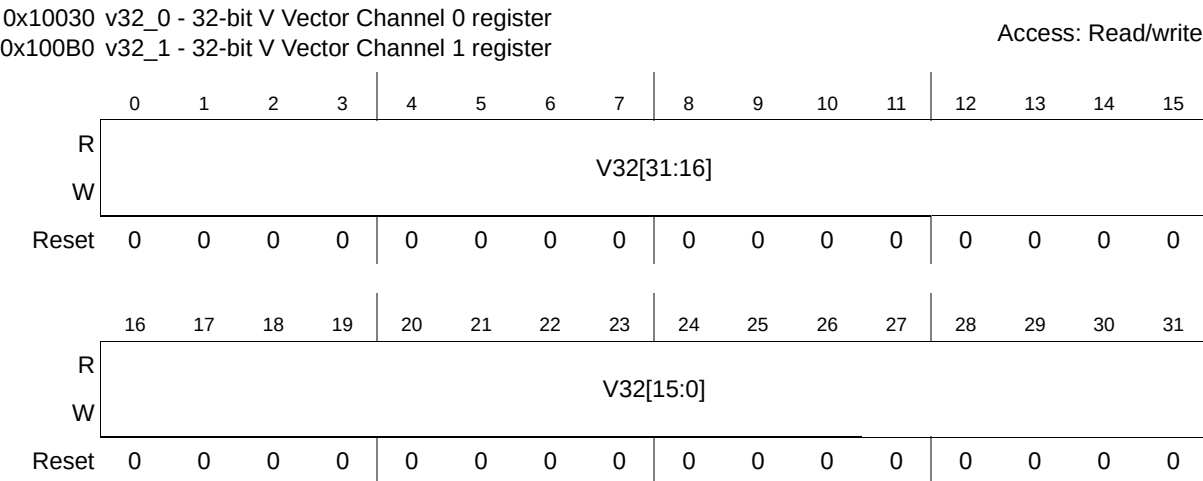


Table 2001. V32_n field description

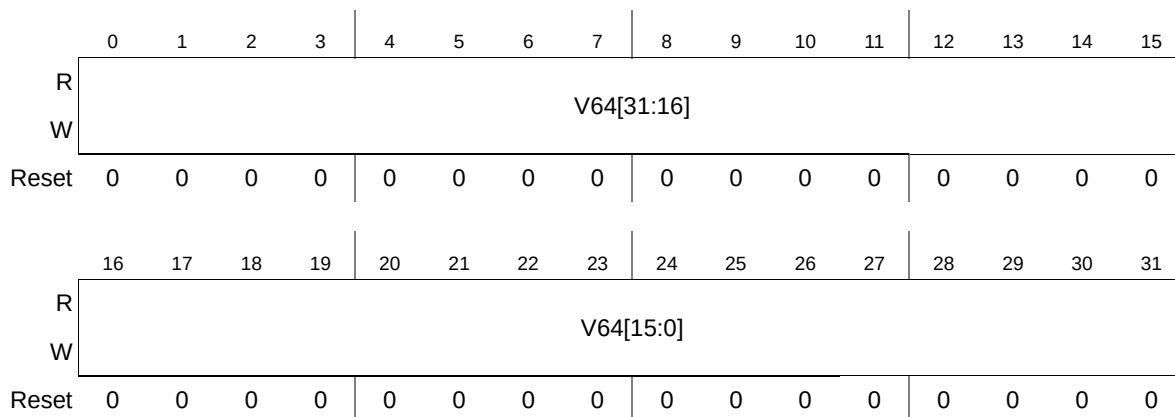
Field	Description
0:31 V32[31:0]	32-bit v-vector, IEEE 32-bit floating point format. When a value is written to the register AMU_v32, the register AMU_v64 is also updated. The conversion from 32-bit IEEE floating point to 64-bit IEEE floating point is done on-the-fly. When a processor writes to the register AMU_v64, this register is not updated. When the calculation engine advances, and increments p8, the new V value is written both to AMU_v64 and AMU_v32.

86.2.2.13 64-bit V Vector Least Significant Bytes Channel *n* register (V64Ln)

Figure 2144. 64-bit V Vector Least Significant Bytes Channel *n* register

0x10034 V64L0 - 64-bit V Vector Least Significant Bytes Channel 0 register
 0x100B4 V64L1 - 64-bit V Vector Least Significant Bytes Channel 1 register

Access: Read/write



86.2.2.14 64-bit V Vector Most Significant Bytes Channel *n* register (V64Mn)

Figure 2145. 64-bit V Vector Most Significant Bytes Channel *n* register (V64Mn)

0x10038 V64M0 - 64-bit V Vector Most Significant Bytes Channel 0 register
 0x100B8 V64M1 - 64-bit V Vector Most Significant Bytes Channel 1 register

Access: Read/write

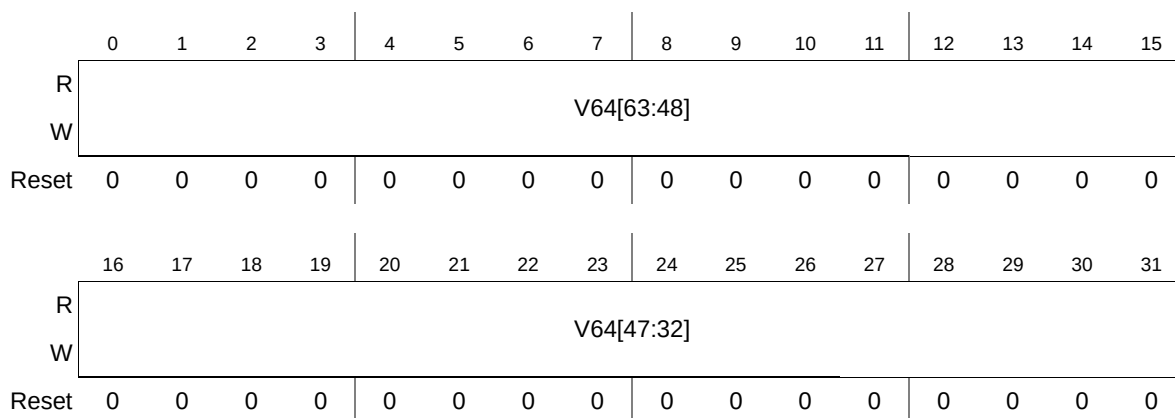


Table 2002. V64Mn field description

Field	Description
v64[63:0]	v parameter 64 bit IEEE floating-point notation. Note: writing to v32 also updates v64 with the same floating-point value, but in 64-bit floating point notation.

86.2.2.15 AMU DMA Source Address Channel *n* register (DSACH_{*n*})Figure 2146. DMA Source Address Channel *n* register (DSACH_{*n*})

0x1003C DSACH0 - DMA Source Address Channel 0 register

0x100BC DSACH1 - DMA Source Address Channel 1 register

Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SADDR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADDR[15:3]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2003. DSACH_{*n*} field description

Field	Description
0:28 SADDR[31:3]	DMA source address in flash. dma_source_addr[31:0] is a byte address. 3 LSBs are forced 0, causing double word alignment. This address is auto-incremented during DMA operation - do no write while DMA is running
29:31	Reserved.

86.2.2.16 AMU DMA Destination Address Channel *n* register (DDACH_{*n*})Figure 2147. DMA Destination Address Channel *n* register (DDACH_{*n*})

0x10040 DDACH0 - DMA Destination Address Channel 0 register

0x100C0 DDACH1 - DMA Destination Address - Channel 1 register

Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DADDR[31:3]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2004. DDACHn field description

Field	Description
0:15	Reserved.
16:28 DADDR[15:3]	DMA destination address. dma_dest_address[15:0] is a byte address. As 3 LSBs are forced zero, double word alignment is forced. This address is auto-incremented during DMA operation - do not write while DMA is running.
31 CADDR	Addressing modes. 0: Incrementing addressing on dma_daddr 1: Circular addressing modulo 256 bytes on dma_daddr

86.2.2.17 AMU DMA Transfer Count Channel *n* register (DCNTCHn)Figure 2148. DMA Transfer Count Channel *n* register (DCNTCHn)

0x10044 DCNTCH0 - DMA Transfer Count Channel 0 register

0x100C4 DCNTCH1 - DMA Transfer Count Channel 1 register

Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT[15:3]													0	0	0
W														0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2005. DCNTCHn field description

Field	Description
0:15	Reserved.
16:28 COUNT[15:3]	DMA transfer count. dma_count[15:0] is a byte count. As 3 LSBs are zero, count is forced in increments of double words. This register is updated (decremented) while the DMA is running. Do not write while DMA is busy.
29:31	Reserved.

86.2.2.18 AMU DMA Transfer Control Channel *n* register (DCTLCH*n*)Figure 2149. DMA Transfer Control Channel *n* register (DCTLCH*n*)

0x10048 DCTLCH0 - DMA Transfer Control Channel 0 register

Access: Read/write

0x100C8 DCTLCH1 - DMA Transfer Control Channel 1 register

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SRT	STP	WVS		DINT	DIE	SKIP	DFA	AP[15:8]							
W				WVC												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIN[7:0]								MAX[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2006. DCTLCH*n* field description

Field	Description
0 SRT	DMA transfer in progress 0: No action 1: DMA transfer start request
1 STP	Result of last stop action – Read 0: transfer was completed - stop did not have effect 1: transfer was in progress - stop did cancel transfer – Write 0: No action 1: DMA transfer stop request
2 WVS	– Wait if VSRT = 1 – Read 0: DMA operates normally 1: DMA does not move data while VSRT set (AMU Status Configuration Register flag) Set on WVS=1(read mode) Cleared on WVC written 1, or VSRT cleared – Write 0: No action 1: Set WVS
3 WVC	WVS clear 0: No action 1: Clear WVS

Table 2006. DCTLCHn field description (continued)

Field	Description
4 DINT	Done interrupt flag Read Set on completion of DMA transfer Cleared on writing '1' to DIN Write Done Interrupt Clear. 0: No action 1: Clear DINT
5 DIE	Done Interrupt Enable 0: Interrupt disabled 1: Interrupt enabled
6 SKIP	0: normal operation 1: the first $p8 * p7$ halfwords or words (depending on v16 bit setting) are not transferred. However, counters are incremented as if these bytes have been transferred
7 DFA	DMA FIFO 0: Normal operation 1: DMA FIFO active. Needs to be used together with circular addressing on the destination pointer. In this case, the FIFO logic stalls the DMA when an overflow of the circular buffer is imminent.
8:15 AP[15:8]	Averaging period This is a time in increments of 256 clocks.
16:23 MIN[7:0]	Averaged over the av_period, the DMA does not transfer more than one 32-bit packet per this number of clock periods.
24:31 MAX[7:0]	The DMA transfers at least one 32-byte packet per this number of clocks. If less data is transferred, the DMA activates high-priority mode on its private flash interface. The transfer time is averaged over the complete length of the DMA transfer.

86.2.2.19 AMU Status and Configuration register (STCFGn)

This section provides details on the AMU Status and Configuration registers (STCFGn).

Figure 2150. AMU Status and Configuration register (STCFGn)

0x1004C STCFG0 - Status and Configuration register

0x100CC STCFG1- Status and Configuration register

Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	USRT	VSRT	VSTP	WDMA		VINT	OINT	EINT	NINT							
W					WD MAC											DFA
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R						VIE	OIE	EIE	NIE		ASTOP	EON	VOOE	VSO	VSN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2007. STCFGn field description

Field	Description
0 USRT	U Calculation Start and Status The U calculation can be initiated by writing this bit to 1. The bit reads as 1 while the U calculation is in progress and reads as 0 after it completes.
1 VSRT	V Calculation Start and Status The V calculation can be initiated by writing this bit to 1. The bit reads as 1 while the V calculation is in progress and reads as 0 after it completes or is stopped.
2 VSTP	V Stop Request and Status – Read 0: Calculation was not underway when VSTP request was made. VSTP did not have effect. 1: Calculation was stopped by VSTP request, result is not final, calculation needs to be restarted. Intermediate V32 values and the P8 loop index are consistent and valid upon stopping a calculation. VSTP is also asserted in response to a properly enabled overflow or Ncrossing stop event. – Write 0: No action 1: Request to stop V calculation

Table 2007. STCFGn field description (continued)

Field	Description
3 WDMA	– Read 0: No delay on start of V-calculation. 1: V-calculation is postponed until DMA done or DMA stop reached. - Set on write to WDMA, cleared on write to WDMAC or DMA completion – Write 0: No action 1: Set WDMA
4 WDMAC	0: No action 1: Clear WDMA
5 VINT	- V Calculation Complete Interrupt. – Read - Set on completion of V calculation. – Write 0: No action 1: Clear V Calculation Complete Interrupt
6 OINT	- Overflow interrupt – Read. - Set on overflow on final result or V.[– Write 0: No action 1: Clear Overflow interrupt
7 EINT	- Error Interrupt. – Read - Set on error detect – Write 0: No Action 1: Clear EINT
8 NINT	- Ncrossing interrupt. – Read - Set when NCRS == (p8-1) and p8 is incremented. The final value of p8 after the interrupt is NCRS. – Write 0: No action 1: Clear Ncrossing Interrupt
21 VIE	- V calculation Interrupt Enable - Controls if VINT is sent to interrupt controller or not. 0: V Calc Interrupt disabled 1: V Calc Interrupt enabled
22 OIE	- Overflow Interrupt Enable 0: Overflow interrupt disabled 1: Overflow interrupt is enabled

Table 2007. STCFGn field description (continued)

Field	Description
23 EIE	Error Interrupt Enable. 0: Error interrupt is disabled 1: Error interrupt is enabled
24 NIE	Ncrossing Interrupt Enable 0: Ncrossing interrupt disabled 1: Ncrossing interrupt enabled
27 EON	Exponential on Negative (Infinity). 0: Generates zero on its output when input value is negative infinity 1: Exponential function generates infinity on its output when input value is negative infinity
28 VOOE	V operand overflow 0: No overflow signalling on V. Only on F 1: Overflow interrupt is triggered on V operand overflow
29 VSO	0: V calculation continues on overflow 1: Vstop automatically triggers on V overflow
30 VSN	0: Ncrossing does not influence Vstop. 1: Vstop automatically triggers on Ncrossing interrupt
15 DFA	DMA FIFO Active. 0: DMA FIFO is not active. Normal operation 1: DMA FIFO is active. Allows start of calculation to be triggered before DMA transfer for V vector completed.
26 ASTOP	DMA Auto Stop 0: DMA stop not triggered 1: DMA stop triggered on any Vstop condition

86.2.2.20 Error Status Register (ESRn/ESRnNC)

This section provides details on the AMU Error Status registers (ESR) and Error Status registers No Clear on read (ESRnNC).

Figure 2151. Error Status Register (ESR)

0x10050 ESR0 error flags, auto clear, channel 0

0x10054 ESR0NC error flags, no clear, channel 0

0x100D0 ESR1 error flags, auto clear, channel 1

0x100D4 ESR1NC error flags, no clear, channel 1

Access: Read only

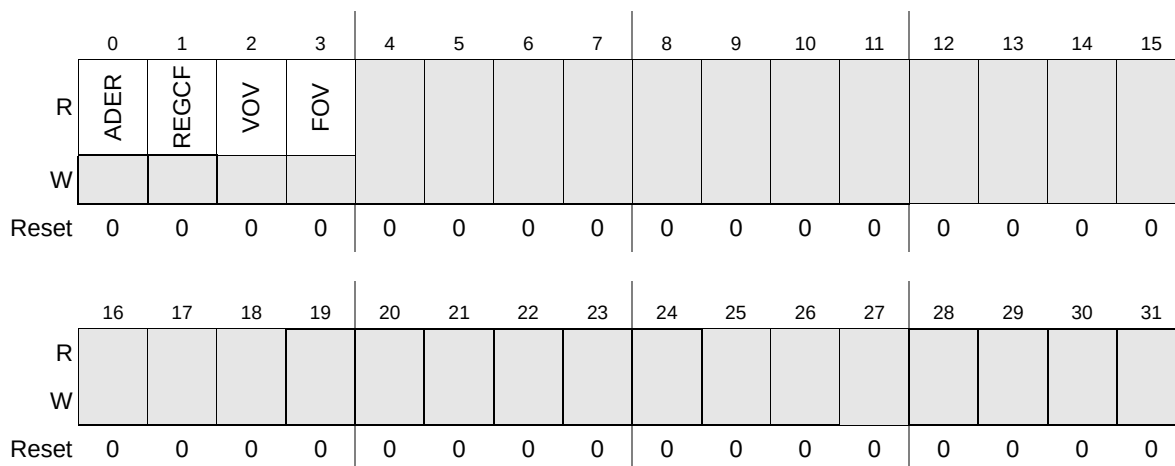


Table 2008. ESR field description

Field	Description
0 ADER	Address Error. Read or write to unimplemented address of the local RAM.
1 REGCF	Register Conflict Error. While V calculation is in progress for a given channel, one of the following registers associated with the same channel was written: 1. RESCHn (result) 2. V32_n 3. V64M_n 4. V64L_n 5. P8CHn
2 VOV	Overflow on intermediate V32_n value.
3 FOV	Overflow on final result RESCHn.
4:31	Reserved.

Reading ESR0 or ESR0NC returns the channel 0 error status flags. Reading ESR1 or ESR1NC returns the channel 1 error status flags. The channel 0 error status flags are automatically cleared on a read from ESR0, but not on read from ESR0NC. The channel 1 error status flags are automatically cleared on read from ESR1, but not on read from ESR1NC.

86.2.2.21 AMU DMA V Vector Holdoff Count Channel *n* register (DVHCH_{*n*})

Figure 2152. DMA V Vector Holdoff Count Channel *n* register (DVHCH_{*n*})

0x10058 DVHCH0 - DMA V Vector Holdoff Count Channel 0 register

0x100D8 DVHCH1 - DMA V Vector Holdoff Count Channel 1 register

Access: Read/write

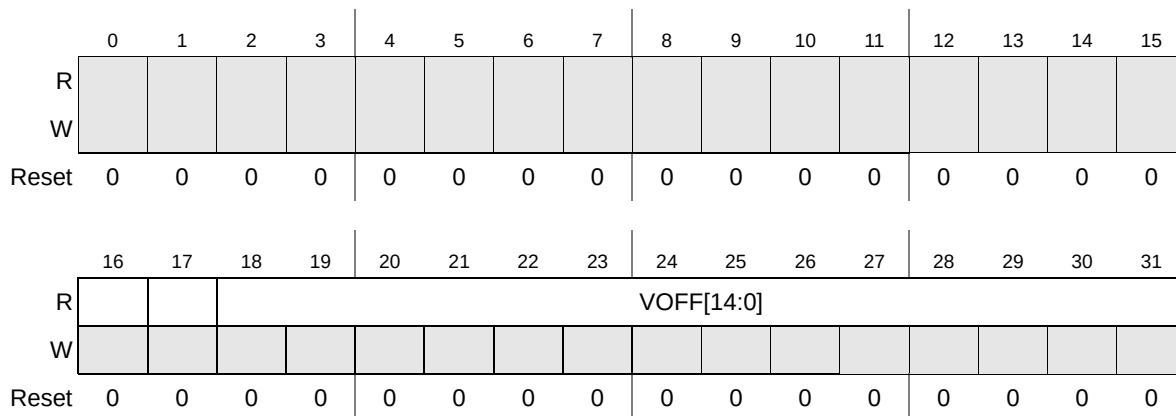


Table 2009. DVHCH_{*n*} field description

Field	Description
0:16	Reserved.
18:31 VOFF[14:0]	Vector holdoff count. Value equal to p7*p8 Number of elements of the V-vector the DMA does not transfer if DCTLCH _{<i>n</i>} [SKIP] = 1

86.3 Functional description

The AMU block has two independent channels, called CHANNEL 0 and CHANNEL 1, featuring independent registers. Each of the AMU channels has a calculation engine, and a DMA. The calculation engine supports algorithmic calculations based on the data in the AMU's local RAM, and visible over the peripheral bus. The DMA executes data transfers from the system flash to the AMU local RAM using a private interconnect. The calculation engine has two start bits, STCFG_{*n*}[USRT, VSRT], that each initiate part of a calculation to be performed.

1. When STCFG_{*n*}[USRT] is set, it activates the updating of the U vector:

```
int k ;
double t ;
extern float *U_pointer ;    // register
extern float * p1_pointer ; // register
extern float * p2_pointer ; // register
extern short int p7 ;        // register 5-bit int p7
for(k = 0; k < p7; k=k+1)
{
    *(U_pointer +k) = *(U_pointer+k) * *(p2_pointer+k) -
                      *(p1_pointer+k) * *(p2_pointer+k) ;
}
```

```

t = 0 ;
for(k = 0; k < p7; k=k+1)
{
    t = t - *(U_pointer+k) * *(U_pointer+k) ;
}
result = t;

```

2. When STCFGn[VSRT] is set, it activates the updating of the v32, v64, p8 and result values in the corresponding registers:

```

int k ;
double t ;
extern short int p6 ;      // register
extern short int p8 ;      // register
extern short int p7 ;      // register, 5-bit int p7
extern float *U_pointer ;  // register
extern float *V_pointer ;  // register
                           // if V16 set, V-pointer is pointer to 16-bit
float.
extern float *p3_pointer ; // register
extern float v32 ;         // register
extern double v64 ;        // register
extern float p4 ;          // register
extern float p5 ;          // register
extern float result ;      // register
extern float exp(double t); // exp function takes double argument,
returns float.

while(p8 < p6)
{
    t = result ; // this is the result of the U-vector calculation
    for(k = 0; k < p7; k=k+1)
    {
        t = t + *(V_pointer + p8*p7+k) * *(U_pointer+k) ;
    }
    v64 = v64 + *(p3_pointer+j) * exp(t) ;
    v32 = (float) v64 ;
    p8 = p8 + 1 ;
}
result = v32 * p4 + p5 ;

```

The first part of the calculation is not interruptible, whereas the second part is. There are stop flags controlling the interruption, and there is an external input signal (ipg_debug) that can stall the calculation.

If both calculations need to be done in sequence, it suffices to set both start flags in a single write to the STCFGn register. The hardware ensures the calculations are completed in the proper order.

86.3.1 Register data types

In the pseudo-code given above, some values are taken from registers. Their data types are defined in [Table 2010](#).

Table 2010. Data types in registers

Operand	Type
U_pointer ⁽¹⁾	Pointer to IEEE 32-bit floating-point
p1_pointer	Pointer to IEEE 32-bit floating-point
p2_pointer	Pointer to IEEE 32-bit floating-point
result	IEEE 32-bit floating point
p6	Integer with limited number of bits
p8	Integer with limited number of bits
p7	Integer with limited number of bits
p3_pointer	Pointer to IEEE 32-bit floating-point
v_pointer	Pointer to IEEE 32-bit floating-point or Pointer to IEEE 16-bit floating-point
v32	IEEE 32-bit floating-point
v64	IEEE 64-bit floating-point
p4	IEEE 32-bit floating-point
p5	IEEE 32-bit floating-point

1. All pointers reference the AMU local RAM, and are limited to 16-bits wide as the RAM is maximum 64 KB.

86.3.2 Calculation precision—Deviation from IEEE standard

- All multiplies are done with the precision of their input operands, into an IEEE 64-bit floating point number (double).
- All additions are done with IEEE 64-bit floating-point precision.
- There is an error of one LSB on the 64-bit v64 register when the value is negative.
- Rounding is always to nearest.
- Denorm numbers and NAN are not supported. They are interpreted as zero and infinity respectively.
- Exponential function inputs a 64-bit *t* operand and produces an IEEE 32-bit floating point result. The precision of the result is 23 mantissa bits (including the 'hidden' bit).

86.3.3 DMA

Each channel is equipped with one DMA channel. This allows optimized copying of data from flash and system calibration memories (if applicable) to local RAM. There is a source address, a destination address, and a count register. All three registers are updated while the DMA is running. The DMA terminates when the count decrements to zero.

86.3.4 Advanced features

86.3.4.1 DMA FIFO

There is a coupling mechanism implemented between the Calculation Engine and the DMA, in the form of a data FIFO. This mechanism allows the AMU calculation engine to be started before the DMA transfer for the corresponding V vector completes. To activate the coupling mechanism, set the DCTLCHn[DFA] and STCFGn[DFA] bits. Next start the DMA by setting DCTLCHn[SRT], and without waiting for the DMA transfer to complete^(z), start the AMU calculation engine.

The mechanism stalls the AMU calculation engine when the DMA is lagging, or it stalls the DMA if it gets too far ahead of the AMU calculation engine.

When the DMA FIFO is used, the DMA source pointer initially points to V[0], the first element of the V-vector in flash. The DMA destination pointer defines the base of the V-vector in local RAM. This is the same value as programmed in the AMU V_pointer. The DMA count is set equal to the total size (in bytes) of the V-vector. Its value needs to be $p6 * p7 * 2$ (in case of 16-bit V-vector) or $p6 * p7 * 4$ (in case of 32-bit V-vector).

As such, the DMA is programmed to transfer the V vector values, starting from V[0]. Now, if the AMU calculation engine p8 value is nonzero, the values V[0: (p8 * p7-1)] are transferred, but not used by the calculation engine.

If the DCTLCHn[SKIP] bit is set, the DMA suppresses, that is, skips, the transfer of the not-used V-vector values. When the DMA transfer is started, the AMU calculation engine examines the values of p8, p7 and V16, and skips any part of the DMA transfer that is not needed by the AMU calculation engine^(aa).

The number of skipped elements, which is equal to $p8 * p7$ (bytes), is available in the DMA V Vector Holdoff Count Channel n register.

86.3.4.2 Circular transfer

When DMA FIFO feature is used^(ab), it is possible to activate circular transfer (modulo 256) on the V pointer and on the daddr pointer. This limits the V-space used in local RAM to 256 bytes. It needs to be used on both pointers in conjunction.

86.3.4.3 Flow control

In order to prevent excessive dissipation by the AMU, and in order to prevent excessive flash bandwidth usage, an effective mechanism for flash bandwidth throttling is implemented^(ac). There are three bit fields for this function in the DCTLCHn register: AP[15:8], MIN[7:0] and MAX[7:0]. AP[15:8] defines the averaging period for max bandwidth while MIN[7:0] effectively forces a minimum time between AMU memory requests. The mechanism does not immediately limit bandwidth; it makes sure that averaged over

z. The DMA transfer does not need to be complete. However, it does not harm if the AMU calculation start is delayed (for example, the processor is servicing an interrupt) until the DMA transfer completes.

aa. If DCTLCHn[SKIP] is set, the registers p7, p8 and the field V16 must have valid values when DMA start is issued (DCTLCHn[SRT] = 1).

ab. The DMA FIFO feature can also be used without the circular transfer feature.

ac. The power dissipation limiting feature is implemented on the AMU DMA, as this function is the major contributor to incremental AMU power dissipation due to its read of flash data.

AP[15:8]×256 clocks, no more than one transfer per MIN[7:0] takes place. MAX[7:0] ensures an average transfer rate (averaged over the duration of the transfer) at least equal to this time. High priority escalation is used to meet this time.

System performance is improved when these times are not equal. This is because processor accesses to flash is very irregular. When they are not equal, the AMU DMA uses all available flash bandwidth that the processors are not using, and when the processors have a flash bandwidth peak, the AMU DMA only activates the high-priority mode when the MAX[7:0] time, averaged over the length of the transfer, is not met any more. So, that means, it catches up when the processors are idle, and wait when the processors are busy, and only activates the high priority mode when the average processor flash access rate is very high.

86.3.4.4 DMA WaitVBusy flag (WVS)

In some use cases, the DMA and the calculation engine operate in an interleaved way. This means, first the DMA does a copy of V-vector data from flash to the local RAM, and when this is done, the calculation engine starts based on this data. When the calculation engine is finished, a new DMA transfer can be started.

The hardware has a flag that allows the DMA to be programmed for the next transfer, while the calculation engine is still running. That means it is possible to program the DMA while the calculation engine is still operating on the previous request. As an immediate DMA start would risk overwriting the current V-vector in local RAM with the newly copied V-vector, in this case, the hardware ensures the DMA start is postponed until the calculation engine is ready. This is done by writing the WVS, in read mode, to one when the early start is given. When the calculation engine is done, the flag automatically clears and the calculation engine starts the next calculation. When the flag is written while the calculation engine is idle, it has no effect.

86.4 Programming examples

86.4.1 AMU operation with U, p1, p2, p3, V vector resident in local RAM

Step 1: Write values for the parameters V pointer, U pointer p1, p2, p3, p8, p6, Ncrossing, p7, p4, p5 to the relevant registers. V16 in V pointer register indicates whether V pointer is an array of 16-bit float else the array is of 32-bit float. Set it to one if V pointer is array of 16-bit float. CADDR bit in V pointer register should be written zero.

Step 2: If there is a 32-bit initial value for V, write it to V32 register. If there is a 64-bit initial value for V, write it to V64 register. One of these registers should be written, not both.

Step 3: Write AMU Status and Configuration register for channel 0.

- Program VSRT and USRT bits equal to 1
- Program DFA equal to 0
- Program VSN equal to 1 if calculation needs to stop on Ncrossing. Program to 0 otherwise

Step 4: Wait for completion (either interrupt, or polling of VSRT flag)

86.4.2 AMU operation with p1, p2, p3, U vector in local RAM, and V vector in flash

Step 1: Write values for the parameters U pointer p1, p2, p3, p8, p6, Ncrossing, p7, p4, p5 to the relevant registers.

Step 2: Write the destination address of the V vector in local SRAM to both V pointer register and AMUDMA destination address register. Write also the V16 bit to V vector bit 2 (indicates 16-bit or 32-bit V vector), and write CADDR bit to both registers. Write '0' for the CADDR bit, if there is enough space in RAM to hold the complete V vector, write '1' for the CADDR bit, if there is only 256 bytes of space in the local RAM. If '0' is written for CADDR, a subsequent AMU operation can read the V vector from local SRAM, if '1' is written for CADDR, the V vector needs to be reread from flash every time its needed.

Step 3: If there is a 32-bit initial value for V, write it to V32 register. If there is a 64-bit initial value for V, write it to V64 register. One of these registers should be written, not both.

Step 4: Write the source address of the V vector in flash to AMUDMA source address. This needs to be the address of V[0], even if p8 is nonzero.

Step 5: Write AMUDMA count equal to $p6 \times p7 \times 4$ if V16 = 1 or $p6 \times p7 \times 2$ if V16=0

Step 6: Write AMUDMA control register.

- Write START = 1
- Write DIE = 1 if DMA done interrupt is enabled, zero otherwise.
- Write SKIP = 1 if values V[0] to V[p8-1] need not be copied from flash to SRAM.
- Write DFA = 1
- Write values for the 3 bandwidth control parameters AP, MIN, MAX

Step 7: Write AMU Status and Configuration register. No need to wait for DMA completion.

- Write VSRT and USRT bits = 1.
- Write DFA = 1
- Write VSN = 1(Vstop on Ncross) if calculation needs to stop on Ncrossing. Program to 0 otherwise.
- Write ASTOP = 1

Step 8: Wait for completion. (either interrupt, or polling of Vstart flag.)

86.4.3 AMU resume

Following are steps required for an AMU resume.

Step 1: Restore registers to values after stop. If no intermediate calculation has been done, no need to restore anything.

Step 2: If this was a chained AMU plus DMA transfer, write start bit in AMUDMA control register to 1.

Step 3: Write VSRT bit in AMU Status and Configuration register.

Step 4: Wait for completion.

86.4.4 DMA only operation

Step 1: Write the destination address of the V vector in SRAM to AMUDMA destination address. Write CADDR equal to 0.

Step 2: Write the source address of the V vector in flash to AMUMDA source address.

Step 3: Write AMUDMA count equal to the number of bytes to be transferred.

Step 4: Write AMUDMA control register.

- Write START = 1
- Write DIE = 1 if DMA done interrupt is enabled, zero otherwise.
- Write SKIP= 0
- Write DFA = 0
- Write values for the 3 bandwidth control parameters AP, MIN, MAX

Step 5: Wait for completion (either interrupt, or polling of VSRT flag)

87 Wakeup Unit (WKPU)

87.1 Introduction

87.1.1 Overview

The Wakeup Unit (WKPU) supports one external source that can cause non-maskable interrupts to on-chip cores and wakeup events to the system. This NMI external source can also generate a functional reset event to the MC_RGM. [Figure 2153](#) is a block diagram of the WKPU and its interfaces to other system components.

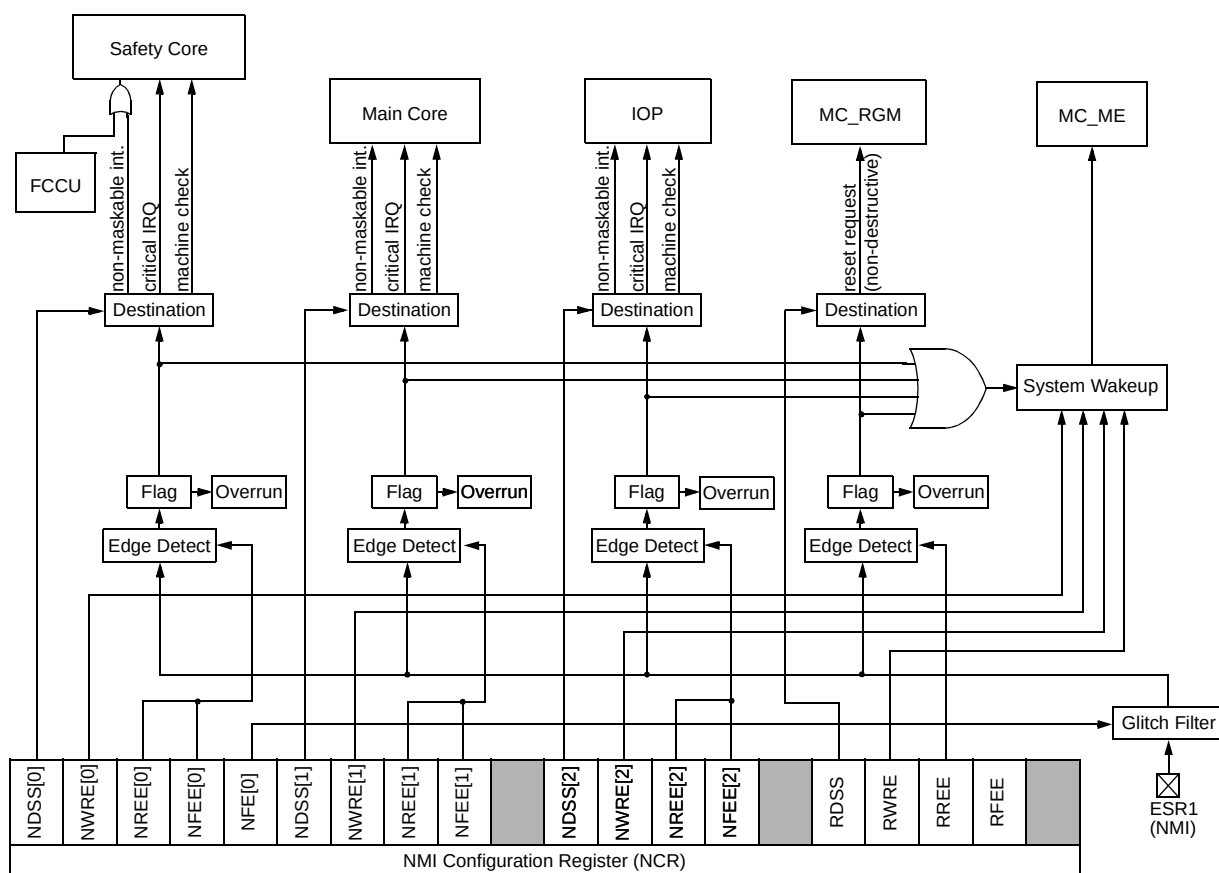


Figure 2153. WKPU block diagram

87.1.2 Features

The WKPU supports these distinctive features:

- one external NMI source
- one analog glitch filter
- independent interrupt destination for each core (safety core, main core, and IOP):
 - non-maskable interrupt
 - critical interrupt
 - machine check request
- active edge selection control for events to each core and functional reset event to MC_RGM
- configurable wakeup event from low power mode to the MC_ME
- configurable wakeup event to exit from STOP mode to the MC_ME
- WKPU register configuration lock

87.2 External signal description

There is a single, active-edge-selectable, NMI pin (ESR1) on the device that connects to the WKPU block. See “Signal Description” chapter for more detail on the ESR1 pin.

87.3 Memory map and register description

This section provides a detailed description of all registers accessible in the WKPU module.

87.3.1 Memory map

[Table 2011](#) gives an overview of the WKPU registers.

Table 2011. WKPU memory map

Address offset	Register	Size (bits)	Supported access sizes (bits)	Location
0x0000	NMI Status Flag Register (NSR)	32	32/16/8	on page 3678
0x0004	Reserved			
0x0008	NMI Configuration Register (NCR)	32	32/16/8	on page 3680
0x000C – 0x03FFF	Reserved			

Note: *Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.*

87.3.2 Register descriptions

This section describes in address order all the WKPU registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

87.3.2.1 NMI Status Flag Register (NSR)

This register holds the status flags of the NMI external source pin and reset request. This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI or reset request event is detected. The overrun flag is set whenever an NMI or reset request event is detected and the status flag is set (i.e. has not yet been cleared). The reset request overrun is a don't-care since the device will be reset on the first event.

The status flags are set independently of the NDSS/RDSS bits. Therefore, these flags can be set but not generate a non-maskable interrupt or system wakeup if NWRE/RWRE = 0 or if NDSS is set to "no request".

Note: The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

Note: NIF is set when: NMI Destination Source Select is disabled (NDSS = 11) and NMI System Wakeup Request is disabled (NWRE = 0) and an NMI input event is seen (pad4 is toggled).

Also see notes in [Section 87.3.2.2, NMI Configuration Register \(NCR\)](#).

Figure 2154. NMI Status Flag Register (NSR)

Address: 0x0000

Access: User read/write (write 1 to clear)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NIF[0]	NOVF[0]	0	0	0	0	0	0	NIF[1]	NOVF[1]	0	0	0	0	0	0
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIF[2]	NOVF[2]	0	0	0	0	0	0	RIF	ROVF	0	0	0	0	0	0
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2012. NSR field descriptions

Field	Description
NIF _[x⁽¹⁾]	NMI Status Flag x This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE[x] or NFEE[x] set), an NMI input event will set NIF[x]. NIF[x] causes an interrupt request based on the NMI destination, as configured by the NDSS bits, and/or a system wakeup request based on the NWRE bit configuration. 1An event as defined by NREE[x] and NFEE[x] has occurred 0 No event has occurred on the pad
NOVF _[x⁽¹⁾]	NMI Overrun Status Flag x This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF[x] value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. NOVF[x] causes an interrupt request based on the NMI destination, as configured by the NDSS bits, and/or a system wakeup request based on the NWRE bit configuration. 1An overrun has occurred on NMI input x 0 No overrun has occurred on NMI input x
RIF	Reset status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (i.e., if RREE or RFEE is set), RIF status bit is set and will cause a functional reset request to RGM if RDSS[1:0] of NCR is '00'. This bit is automatically cleared if the ESR1 reset request is configured in the MC_RGM to generate a long or short "functional" reset. 1An event as defined by RREE and RFEE has occurred 0No event has occurred on the pad
ROVF	Reset overrun status flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current RIF value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (RREE or RFEE set), ROVF can cause a reset request to MC_RGM if RDSS[1:0] of NCR is '00'. This bit is automatically cleared if the ESR1 reset request is configured in the MC_RGM to generate a long or short "functional" reset. 1An overrun has occurred on NMI input 0No overrun has occurred on NMI input

1. Index 'x' designates the core: 0 is the safety core, 1 is the main core, and 2 is the IOP.

87.3.2.2 NMI Configuration Register (NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

Figure 2155. NMI Configuration Register (NCR)

Address: 0x0008

Access: User read/write

	0	1	2	3	4	5	6	7
R	NLOCK[0]	NDSS[0]	NWRE[0]	0	NREE[0]	NFEE[0]	NFE[0]	
W								
Reset	0	0	0	0	0	0	0	0
	8	9	10	11	12	13	14	15
R	NLOCK[1]	NDSS[1]	NWRE[1]	0	NREE[1]	NFEE[1]		0
W								
Reset	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23
R	NLOCK[2]	NDSS[2]	NWRE[2]	0	NREE[2]	NFEE[2]		0
W								
Reset	0	0	0	0	0	0	0	0
	24	25	26	27	28	29	30	31
R	RLOCK	RDSS	RWRE	0	RREE	RFEE		0
W								
Reset	0	1	1	0	0	0	0	0

Table 2013. NCR field descriptions

Field	Description
NLOCK[x ⁽¹⁾]	NMI Configuration Lock Register x Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.
NDSS[x ⁽¹⁾]	NMI Destination Source Select x 00 Non-maskable interrupt. A non-maskable interrupt will occur if NIF[x] or NOV[x] is set. 01 Critical interrupt. A critical interrupt will occur if NIF[x] or NOV[x] is set. 10 Machine check request. A machine check request will occur if NIF[x] or NOV[x] is set. 11 No reaction
NWRE[x ⁽¹⁾]	NMI System Wakeup Request Enable x 1 A set NIF[x] bit or set NOV[x] bit causes a system wakeup request 0 System wakeup requests from the corresponding NIF[x] bit are disabled
NREE[x ⁽¹⁾]	NMI Rising-edge Events Enable x 1 Rising-edge event is enabled 0 Rising-edge event is disabled

Table 2013. NCR field descriptions (continued)

Field	Description
NFEE[x ⁽¹⁾]	NMI Falling-edge Events Enable x 1 Falling-edge event is enabled 0 Falling-edge event is disabled
NFE[0]	NMI Filter Enable Enable analog glitch filter on the NMI pad input. 1 Filter is enabled 0 Filter is disabled
RLOCK	Reset Configuration Lock Register Writing a 1 to this bit locks the configuration for the reset until it is unlocked by a system reset. Writing a 0 has no effect.
RDSS	Reset Destination Source Select 00 Reset request to RGM 01 No reaction 10 No reaction 11 No reaction
RWRE	Reset Wakeup Request Enable 1 A set RIF bit or set ROVF bit causes a system wakeup request 0 System wakeup requests from the corresponding RIF bit are disabled
RREE	Reset Rising-edge Events Enable 1 Rising-edge event is enabled 0 Rising-edge event is disabled
RFEE	Reset Falling-edge Events Enable 1 Falling-edge event is enabled 0 Falling-edge event is disabled

1. Index 'x' designates the core: 0 is the safety core, 1 is the main core, and 2 is the IOP.

Note: Writing a '0' to both NREE[x] and NFEE[x] disables the NMI functionality completely (i.e. no system wakeup or interrupt will be generated on any pad activity)!

Note: Writing to the NCR when NIF = 1 or NOV = 1 causes an interrupt to the destination source/system wakeup to be generated independent of an NMI wakeup event.

87.4 Functional description

This section provides a complete functional description of the features of the WKPU.

87.4.1 Non-maskable interrupts

The WKPU allows the external NMI pin (ESR1) to assert the non-maskable interrupts of the cores on the device. These interrupts include non-maskable interrupts, critical interrupts, or machine check requests to any of the cores including the main core, IOP, and safety core. The safety core's non-maskable interrupt line from the WKPU is combined with the FCCU internal NMI interrupt event arising due to fault management logic.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See [Table 2013](#) for details.

The enabling of rising edge, falling edge, or either edge reaction to the NMI pin is independent for each core, and is controlled by NFEE[2:0] and NREE[2:0] bits in the NCR.

The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

The NMI supports a status flag and an overrun flag, which are located in the NSR. See [Section 87.3.2.1, NMI Status Flag Register \(NSR\)](#) for more information.

87.4.2 Reset management

The WKPU allows the external NMI pin (ESR1) to assert a non-destructive reset request to the MC_RGM block. The ESR1 reset request can be reconfigured in the MC_RGM to generate a SAFE mode or interrupt request (see [Chapter 57: Reset Generation Module \(MC_RGM\)](#) for details).

The enabling of rising edge, falling edge, or either-edge reaction to the NMI pin is independent for the reset request, and is controlled by the RFEE/RREE bits in the NCR.

Reset request supports a status flag and an overrun flag, which are located in the NSR. See [Section 87.3.2.1, NMI Status Flag Register \(NSR\)](#) for more information.

87.4.3 System wakeup

The WKPU allows the external NMI pin (ESR1) to assert a system wakeup from STOP to the MC_ME block on the device. The system wakeup enable is asserted in the NCR by any of the NWRE/RWRE bits.

87.4.4 Glitch filter

An analog glitch filter is placed between the NMI pin (ESR1) and the WKPU block. The filter is enabled by the NFE[0] bit in the NCR. See the device data sheet for the glitch filter electrical specifications.

87.4.4.1 Edge detect

Each NMI and reset request can be enabled or disabled independently. This can be performed using the single NCR laid out to contain all configuration bits for a given NMI in a single byte (see [Figure 2155](#)). The user can configure interrupts with an active rising edge, an active falling edge, or either edge being active. Disabling both edge events results in no interrupt or reset being propagated from the WKPU block.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits. The active reset request edge is controlled by the RREE and RFEE bits.

Note: After reset, NREE/RREE and NFEE/RFEE are set to '0', therefore the NMI functionality and reset request is disabled after reset and must be enabled explicitly by software.

87.5 Initialization information

This section discusses initialization for the following features:

- glitch filter and pad configuration
- non-maskable interrupt initialization
- reset request

87.5.1 Glitch filter and pad configuration

Glitch filter control and pad configuration should be performed while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

Note: When enabling the glitch filter, do not enable the rising-/falling-edge events bits (NREE[x], NFEE[x], RREE, RFEE) in the same register write.

87.5.2 Non-maskable interrupt initialization

When an NMI interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization the user must do the following:

1. mask interrupts by clearing the NIF[n] and NOV[n] bits in WKPU_NSR
2. clear WKPU_NCR bit NFEE[n] and NREE[n], preferably by writing 0 to all bits in this register
3. enable the analog glitch filter as desired by writing the NFE[0] bit in the WKPU_NCR
4. configure the appropriate SIUL2_MSCR for the desired NMI interrupt pin(s) as follows:
 - clear the ODC bits to disable output
 - set the IBE bit to enable the pin's input buffer
 - if the internal weak pull-up/pull-down is used, configure the appropriate WPUE and WPDE bits

Note: NMI interrupt pins should never be configured as outputs (MSCR_ODC bits are not zeros) when external interrupt inputs are desired since false interrupts could be detected (such as from a GPIO configuration).

5. write appropriate bit values to the WKPU_NCR to configure:
 - NMI source (NDSS[n])
 - wakeup request (NWRE[n])
 - glitch filter (NFE[0])
 - rising/falling edge events enable (NFEE[n] and NREE[n])
 - NMI configuration lock (NLOCK[n])

87.5.3 Reset request

When an NMI reset request pin is first enabled, it is possible to get a false reset request. To prevent a false reset request during pin reset request initialization the user must do the following:

1. mask reset requests by clearing the RIF and ROVF bits in WKPU_NSR
2. clear WKPU_NCR bit RFEE and RREE, preferably by writing 0 to all bits in this register
3. enable the analog glitch filter as desired by writing the NFE[0] bit in the WKPU_NCR
4. configure the appropriate SIUL2_MSCR for the NMI reset request pin(s) as follows:
 - clear the ODC bits to disable output
 - set the IBE bit to enable the pin's input buffer
 - if the internal weak pull-up/pull-down is used, configure the appropriate WPUE and WPDE bits

Note: NMI reset request pins should never be configured as outputs (MSCR_ODC bits are not zeros) when external reset request inputs are desired since false reset requests could be detected (such as from a GPIO configuration).

5. write appropriate bit values to the WKPU_NCR to configure:
 - reset destination source (RDSS)
 - reset wakeup request (RWRE)
 - glitch filter (NFE[0])
 - rising/falling edge events enable (RFEE and RREE)
 - reset configuration lock (RLOCK)

88 GTM103 Integration (GTMINT) Module

88.1 Introduction

This chapter specifies the functionality of the GTMINT module. The GTMINT is integrated IP with timer and debug functions and is comprised of:

- The Bosch GTM-IP_103 timer module (also called GTM-IP)
- The GTMDI debug interface IP
- 12 ECC error correction encoder / decoders for the RAMs required by the timer.

The timer GTM-IP_103 is a very complex block that can be configured to perform several timer functions. The debugger module is the GTMDI IP (GTM Development Interface IP) that is a Nexus based interface that is connected to the chip level NAR and used to monitor the software and hardware activities of GTM-IP. The ECC sub-blocks generate the parity bits on RAM writes and detect errors in the data read from the RAMs.

Note: This chapter does not intend to present details of GTM-IP nor GTMDI. For full information about the timer and the debugger, please refer to the GTMDI subsection in the 'Calibration and Debug' chapter of the SPC57EM80xx Reference Manual and the GTM103 Reference Manual.

88.1.1 Features

The GTMINT module features include:

- GTM-IP timer features
 - Time base generators
 - Advanced data signal / time base router
 - Input signal processors (32 inputs)
 - Simple output signal generators (48 outputs)
 - Microprocessed output signal generators (40 outputs)
 - Digital PLL
 - BLDC support
 - Safety features - GTM is a non-safety relevant module
 - Output compare unit
 - Monitoring unit
 - CPU bus interface clock operates asynchronously to timer section clock allowing CPU low power operation.
- GTMDI debug block features
 - Full duplex pin interface for medium and high visibility throughput
 - Pin interface shared among all Nexus-based development interfaces on the device
 - Auxiliary Output Port
DATA with MDO (Message Data Out) and MSEO (Message Start/End)
2 EVTO (Event Out) pin for GTM real time signal monitoring
 - Auxiliary Input Port
1 EVTI (Event In) pin

- IEEE 1149.1 (JTAG) Test Access Port (TAP)
Support for optional Multi-JTAG TAP Linking Module (TLM)
4 pins (TDI, TDO, TMS, and TCK)
Reset input TRST driven by either the Nexus Port Controller or an external pin
- GTM Development Support
 - Read/Write registers in debug mode from JTAG port
 - Ability to enter debug mode at reset negation or during normal execution
 - Breakpoint and Watchpoint Configuration
 - Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This allows the development tool to trace reads and writes to selected RAM address ranges
Two data trace windows for each MCS core with programmable address ranges and access attributes are provided. Data trace windowing reduces the requirement on the Aurora port bandwidth by constraining the number of trace locations.
Fetch trace capability controlled by two programmable Watchpoints.
Private messages are used to indicate special MCS core cases not covered by public messages. Up to two^(ad) MCS cores can be debugged at the same time.
 - Watchpoint Messaging (WPM)
- Trigger Control
 - The GTMDI provides selection of trigger signals that will generate debug actions at system level such as Halt GTM itself or cores in the same device.
 - Trace enable signals are provided to control the start/stop of trace messages for each one of the GTM Nexus message sources, such as TIM, TOM, ATOM, etc.
- GTMDI Clock operation
 - The module is able to operate in any frequency ratio considering GTMDI and JTAG frequency. The GTMDI should operate in the same clock source as the GTM IP module.
- ECC encoder / decoder for each of 12 RAM interfaces
 - Parity calculation using the whole word and the target memory address value.
 - Detection and correction of single error on the read data value
 - Detection of single bit error on address field but no need for correction
 - Double error detection on data or address field without error correction.
 - ECC report generation for each RAM containing:
 - Address of the error in the register map (see [Section 88.2, "Memory Map and Register Definition"](#))
 - Non-correctable error indication (not correctable data + address ECC detected errors)

ad. Note that by enabling two MCS cores at the same time for data trace can exceed the capacity of the Nexus Aurora Router throughput thus information can be lost.

- Correctable error indication (correctable data error)
- Generation of 151 interrupt requests and or DMA requests.
- Debugger is allowed to read all GTM registers through the peripheral bus even when the timer is running. Read-access of the debugger is possible without destroying or changing the contents of these registers.
- Debugger is allowed to read all GTM addresses read-/writeable from the peripheral bus when GTM is running or halted.

88.1.2 Modes of Operation

88.1.2.1 Normal Mode

The normal mode is the mode in which the GTMINT module is after a reset is applied. In this mode the timer IP is functional and processes input timing signals and generates timed outputs based on the input timing signals or controlled by the internal channels software. The GTMDI development interface is also active.

88.1.2.2 Stop Mode

This mode is set when the global stop input is received or when the bit field MDIS on the GTMMCR register is asserted. In this case all functional clocks are disabled (ipg_clk and ipg_clk_gtm), but the slave bus clock is not disabled. Please, see more details on [Section 88.3.4, Stop Mode Description](#).

88.1.2.3 Debug Mode (or Halt Mode)

The debug mode or GTM-IP halt mode is externally requested through the GTMDI. In this mode, the GTM-IP main clock is requested to stop (ipg_clk_gtm only) and the bus interface clocks remain running to access all registers and memories. It is possible to read the registers not interfering into the logic status, even for FIFOs, by configuring the debug access controls on GTMDI.

88.1.3 Block Diagram

[Figure 2156](#) shows the interconnections in the GTMINT Module.

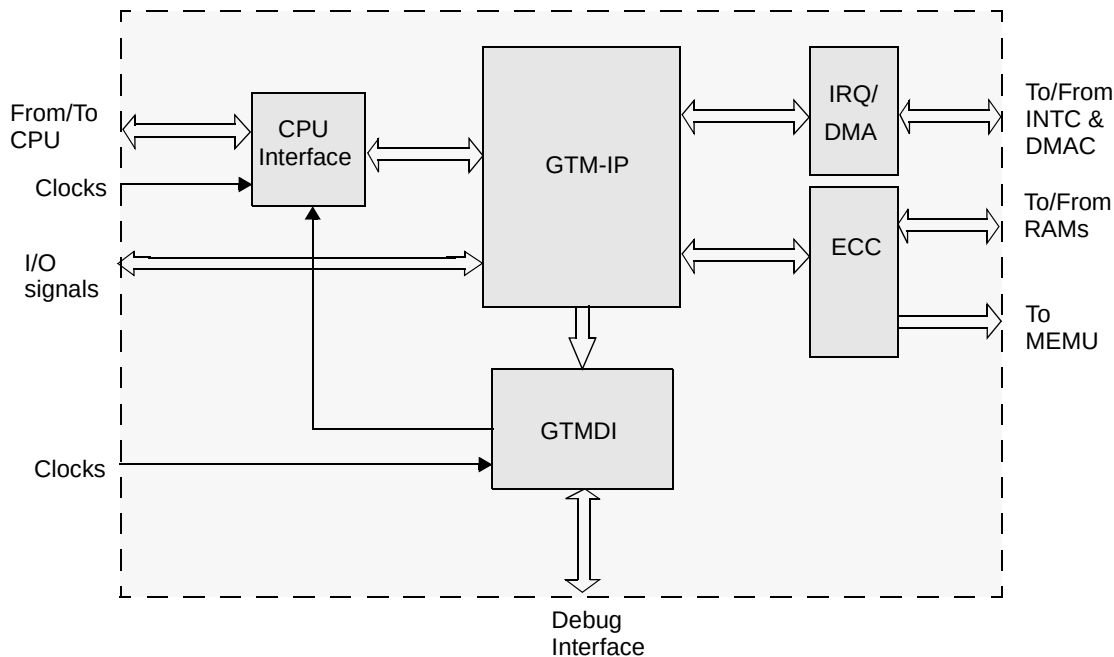


Figure 2156. GTMINT block diagram

88.2 Memory Map and Register Definition

This section provides a reference table to the GTM-IP sub-module registers. Details of each register are found in the *GTM-IP specification* document. Additional registers from those used in the GTM-IP are described in this section.

88.2.1 Module Memory Map

The GTMINT simplified memory map is shown in [Table 2014](#). Offsets are relative to the GTMINT base address.. See the device memory map for the module base address.

Note: The GTMINT module has only a few registers, detailed in [Section 88.2.2, Register Descriptions](#). The module also acts as an interface to all implemented GTM-103 control registers. For details see the GTM-IP Specification document.

Table 2014. High level memory map

Offset	Use
0x00000000	GTM-IP TOP-Level Configuration Registers
0x00000030	GTM-IP AEI Bridge Registers
0x000000C0	GTM-IP Integration Registers
0x00000100	TBU
0x00000180	MON

Table 2014. High level memory map (continued)

0x00000200	CMP
0x00000280	ARU
0x00000300	CMU
0x00000400	BRC
0x00000600	ICM
0x00000800	SPE0
0x00000880	SPE1
0x00000900	SPE2
0x00000980	SPE3
0x00000F00	MAP
0x00000F40	MCFG
0x00001000	TIM0
0x00001800	TIM1
0x00002000	TIM2
0x00002800	TIM3
0x00003000	TIM4
0x00003800	TIM5
0x00004000	TIM6
0x00008000	TOM0
0x00008800	TOM1
0x00009000	TOM2
0x00009800	TOM3
0x0000A000	TOM4
0x0000A800	TOM5
0x0000D000	ATOM0
0x0000D800	ATOM1
0x0000E000	ATOM2
0x0000E800	ATOM3
0x0000F000	ATOM4
0x0000F800	ATOM5
0x00010000	ATOM6
0x00010800	ATOM7
0x00011000	ATOM8
0x00011800	ATOM9
0x00012000	ATOM10
0x00012800	ATOM11

Table 2014. High level memory map (continued)

0x00018000	F2A0
0x00018080	AFD0
0x00018400	FIFO0
0x00019000	FIFO0_MEMORY
0x0001C000	F2A1
0x0001C080	AFD1
0x0001C400	FIFO1
0x0001D000	FIFO1_MEMORY
0x00028000	DPLL
0x00028200	DPLL_MEM R1A
0x00028400	DPLL_MEM R1B
0x00028600	DPLL_MEM_REGION R1C
0x00028E00	DPLL REGION EXT
0x0002C000	DPLL_MEM R2
0x00030000	MCS0
0x00031000	MCS1
0x00032000	MCS2
0x00033000	MCS3
0x00034000	MCS4
0x00035000	MCS5
0x00036000	MCS6
0x00038000	MCS0_MEMORY
0x00040000	MCS1_MEMORY
0x00048000	MCS2_MEMORY
0x00050000	MCS3_MEMORY
0x00058000	MCS4_MEMORY
0x00060000	MCS5_MEMORY
0x00068000	MCS6_MEMORY

This section presents the GTMINT Module Memory Map (that is basically the same of the timer GTM-IP) in [Table 2015](#).

Table 2015. GTMINT Module Memory Map

Offset	Use	Access	Reset value ⁽¹⁾	Location
GTM-IP TOP-Level Configuration Registers				
0x00000	GTM-IP Version control register (GTM_REV)	R	0x1041_51A1	See GTM Spec. ⁽²⁾
0x00004	GTM-IP Global reset register (GTM_RST)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x00008	GTM-IP Global control register (GTM_CTRL)	R/W	0x0000_0001	See GTM Spec. ⁽²⁾
0x0000C	GTM-IP Timeout Address Register (GTM_AEI_ADDR_XPT)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00010	GTM-IP Interrupt Notification Register (GTM_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00014	GTM-IP Interrupt Enable (GTM_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00018	GTM-IP Interrupt Force (GTM_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0001C	GTM-IP Interrupt Mode (GTM_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00020	GTM-IP (GTM_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00024 – 0x0002C	Reserved			
AEI Bridge (BRIDGE) Registers				
0x00030	GTM-IP AEI Bridge Mode (GTM_BRIDGE_MODE)	R/W	0x0400_0001 ⁽³⁾	See GTM Spec. ⁽²⁾
0x00034	GTM-IP AEI Bridge Pointer1 (GTM_BRIDGE_PTR1)	R	0x0040_0000 ⁽³⁾	See GTM Spec. ⁽²⁾
0x00038	GTM-IP AEI Bridge Pointer2 (GTM_BRIDGE_PTR2)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x0003C	Reserved			
TIM AUX Registers				
0x00040+m*0x04	GTM-IP TIM[m] auxiliary source selection register (m = 0..3) (GTM_TIM[m]_AUX_IN_SRC)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00050 – 0x00058	Reserved			
0x0005C – 0x000BC	Reserved			
Top GTM Integration Registers				
0x000C0	GTMMCR – GTM Module Configuration Register	R/W	0xX000_X000 ⁽³⁾	on page 3709
0x000C4	Reserved			
0x000C8	GTMINTCLR – GTM Interrupts Clear Register	R/W	0x0000_0000	on page 3711
0x000CC	GTMMAEICR – GTM AEI Control Register	R/W	0x0000_0000	on page 3713
0x000D0 – 0x000FC	Reserved			
Time Base Unit (TBU) Configuration Registers				
0x00100	TBU global channel enable (TBU_CHEN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00104+m*0x08	TBU channel m control (m = 0..2) (TBU_CH[m]_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x00108+n*0x08	TBU channel n base (n = 0...2) (TBU_CH[n]_BASE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0011C – 0x00120	Reserved for TBU			
0x00124 – 0x0017C	Reserved			
Monitor Unit (MON) Configuration Registers				
0x00180	Monitor Status register (MON_STATUS)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00184	Monitor activity register 0 (MON_ACTIVITY_0)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00188 - 0x001A4	Reserved for MON			
0x001A8 – 0x001FC	Reserved			
Output Compare Unit (CMP) Configuration Registers				
0x00200	Comparator enable register (CMP_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00204	Event notification register (CMP_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00208	Interrupt enable register (CMP_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0020C	Interrupt force register (CMP_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00210	IRQ mode configuration register (CMP_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00214	IRQ mode configuration register (CMP_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00218 – 0x00220	Reserved for CMP			
0x00224 – 0x0027C	Reserved			
Advanced Routing Unit (ARU) Configuration Registers				
0x00280	ARU access register (ARU_ACCESS)	R/W	0x0000_01FE	See GTM Spec. ⁽²⁾
0x00284	ARU access register upper data word (ARU_DATA_H)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00288	ARU access register lower data word (ARU_DATA_L)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0028C	Debug access channel 0 (ARU_DBG_ACCESS0)	R/W	0x0000_01FE	See GTM Spec. ⁽²⁾
0x00290	Debug access 0 transfer register upper data word (ARU_DBG_DATA0_H)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00294	Debug access 0 transfer register lower data word (ARU_DBG_DATA0_L)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00298	Debug access channel 1 (ARU_DBG_ACCESS1)	R/W	0x0000_01FE	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x0029C	Debug access 1 transfer register upper data word (ARU_DBG_DATA1_H)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x002A0	Debug access 1 transfer register lower data word (ARU_DBG_DATA1_L)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x002A4	ARU Interrupt notification register (ARU_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x002A8	ARU Interrupt enable register (ARU_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x002AC	Register for forcing the ARU_NEW_DATA_IRQ interrupt (ARU_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x002B0	IRQ mode configuration register (ARU_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x002B4 – 0x002FC	Reserved			
Clock Management Unit (CMU) Configuration Registers				
0x00300	Clock enable (CMU_CLK_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00304	Global clock control numerator (CMU_GCLK_NUM)	R/W	0x0000_0001	See GTM Spec. ⁽²⁾
0x00308	Global clock control denominator (CMU_GCLK_DEN)	R/W	0x0000_0001	See GTM Spec. ⁽²⁾
0x0030C+n*0x04	Control for clock source n (n = 0...7) (CMU_CLK_[n]_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0032C+m*0x08	External clock m control numerator (m = 0...2) (CMU_ECLK_[m]_NUM)	R/W	0x0000_0001	See GTM Spec. ⁽²⁾
0x00330+m*0x08	External clock m control denominator (m = 0...2) (CMU_ECLK_[m]_DEN)	R/W	0x0000_0001	See GTM Spec. ⁽²⁾
0x00344	Global clock control denominator (CMU_FXCLK_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00348 – 0x003FC	Reserved			
Broadcast Module (BRC) Configuration Registers (n = 0...11)				
0x00400+n*0x08	Read address for input channel n (BRC_SRC_[n]_ADDR)	R/W	0x0000_01FE	See GTM Spec. ⁽²⁾
0x00404+n*0x08	Destination channels for input channel n (BRC_SRC_[n]_DEST)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00460	BRC Interrupt notification register (BRC_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00464	BRC Interrupt enable register (BRC_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x00468	Register for forcing the BRC_DEST_ERR interrupt (BRC_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0046C	IRQ mode configuration register (BRC_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00470	Software reset (BRC_RST)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00474	Software reset (BRC_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00478 – 0x005FC	Reserved			
Interrupt Concentrator Module (ICM) Configuration Registers				
0x00600	ICM Interrupt group register covering infrastructural and safety components (ARU, BRC, AEI, PSM, MAP, CMP, SPE) (ICM_IRQG_0)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00604	ICM Interrupt group register covering DPLL (ICM_IRQG_1)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00608	ICM Interrupt group register covering TIM0, TIM1, TIM2, TIM3 (ICM_IRQG_2)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x0060C	Reserved			
0x00610	ICM Interrupt group register covering MCS0 to MCS3 submodules (ICM_IRQG_4)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00614	Reserved			
0x00618	ICM Interrupt group register covering GTM-IP output submodules TOM0 to TOM1 (ICM_IRQG_6)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x0061C	ICM Interrupt group register covering GTM-IP output submodule TOM2 (ICM_IRQG_7)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00620	Reserved			
0x00624	ICM Interrupt group register covering GTM-IP output submodules ATOM0, ATOM1, ATOM2 and ATOM3 (ICM_IRQG_9)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00628	ICM Interrupt group register covering GTM-IP output submodule ATOM4 (ICM_IRQG_10)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x0062C	Reserved for ICM			
0x00630	ICM Interrupt group register for module error interrupt information (ICM_IRQG_MEI)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00634	ICM Interrupt group register 0 for FIFO channel error interrupt information (ICM_IRQG_CEI0)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00638	ICM Interrupt group register 1 for TIM0..3 channel error interrupt information (ICM_IRQG_CEI1)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x0063C	Reserved for ICM			

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x00640	ICM Interrupt group register 3 for MCS0..3 channel error interrupt information (ICM_IRQG_CEI3)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x00644	Reserved for ICM			
0x00648 – 0x007FC	Reserved			
Sensor Pattern Evaluation (SPE) Registers (AAi = i*0x80, i = 0..1)				
0x00800 + AAi	SPE Control status register (SPE[i]_CTRL_STAT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00804 + AAi	SPE Input pattern definition register (SPE[i]_PAT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00808+n*0x04 + AAi	SPE Output definition registers (SPE[i]_OUT_PAT[n]), n = 0...7	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00828 + AAi	SPE output control register (SPE[i]_OUT_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0082C + AAi	SPE Interrupt notification register (SPE[i]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00830 + AAi	SPE Interrupt enable register (SPE[i]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00834 + AAi	SPE Interrupt generation by software (SPE[i]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00838 + AAi	IRQ mode configuration register (SPE[i]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0083C + AAi	SPE Interrupt enable register (SPE[i]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00840 + AAi	SPE Interrupt generation by software (SPE[i]_REV_CNT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00844 + AAi	IRQ mode configuration register (SPE[i]_REV_CMP)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x00848 – 0x0087C) + AAi	Reserved			
0x00900 – 0x009C0	Reserved for SPE2/3			
0x00A00 – 0x00EFC	Reserved			
TIM0 Input Mapping Module (MAP) Registers				
0x00F00	MAP Control register (MAP_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00F04 – 0x00F3C	Reserved for MAP			
Memory Configuration (MCFG) Configuration Registers				
0x00F40	Memory layout configuration (MCFG_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x00F44 – 0x00FA0	Reserved for MCFG			

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x00FA4 – 0x00FFC	Reserved			
Timer Input Module (TIM) Configuration Registers (BBi = i*0x800, n = 0...7, i = 0..3)				
0x01000+n*0x80 + BBi	Channel n general purpose 0 (TIM[i]_CH[n]_GPR0)	R	0x0X00_0000	See GTM Spec. ⁽²⁾
0x01004+n*0x80 + BBi	Channel n general purpose 1 (TIM[i]_CH[n]_GPR1)	R	0x0X00_0000	See GTM Spec. ⁽²⁾
0x01008+n*0x80 + BBi	Channel n SMU counter (TIM[i]_CH[n]_CNT)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x0100C+n*0x80 + BBi	Channel n SMU counter (TIM[i]_CH[n]_ECNT)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x01010+n*0x80 + BBi	Channel n SMU shadow counter (TIM[i]_CH[n]_CNTS)	R/W	0x0X00_0000	See GTM Spec. ⁽²⁾
0x01014+n*0x80 + BBi	Channel n TDU control (TIM[i]_CH[n]_TDUC)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x01018+n*0x80 + BBi	Channel n TDU control (TIM[i]_CH[n]_TDUV)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0101C+n*0x80 + BBi	Channel n filter parameter 0 (TIM[i]_CH[n]_FLT_RE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x01020+n*0x80 + BBi	Channel n filter parameter 1 (TIM[i]_CH[n]_FLT_FE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x01024+n*0x80 + BBi	Channel n control (TIM[i]_CH[n]_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x01028+n*0x80 + BBi	Channel n control (TIM[i]_CH[n]_ECTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0102C+n*0x80 + BBi	Channel n interrupt notification (TIM[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x01030+n*0x80 + BBi	Channel n interrupt enable (TIM[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x01034+n*0x80 + BBi	Channel n software interrupt force (TIM[i]_CH[n]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x01038+n*0x80 + BBi	Channel n IRQ mode configuration register (TIM[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0103C+n*0x80 + BBi	Channel n software interrupt force (TIM[i]_CH[n]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x01040+n*0x80 – 0x01074+n*0x80) + BBi	Reserved for TIM[i]_CH[n]			
0x01078 + BBi	Channel n IRQ mode configuration register (TIM[i]_IN_SRC)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x107C + BBi	TIM Global software reset (TIM[i]_RST)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
(0x01078+n*0x80 – 0x0107C+n*0x80) + BBi	Reserved (n > 0)			
(0x01400 – 0x017FC) + BBi	Reserved			
0x03000 – 0x04400	Reserved for TIM4 to TIM6			
0x04404 – 0x07FFC	Reserved			
Timer Output Module (TOM) Configuration Registers (CCi = i*0x800, i = 0...2, n = 0...15)				
0x08000 + n*0x0040 + CCi	TOM Channel m control register (TOM[i]_CH[n]_CTRL), i = 0...2, n = 0...15	R/W	0x0000_X000	See GTM Spec. ⁽²⁾
0x08004 + n*0x0040 + CCi	TOM Channel n CCU0 compare shadow register (TOM[i]_CH[n]_SR0)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08008 + n*0x0040 + CCi	TOM Channel n CCU1 compare shadow register (TOM[i]_CH[n]_SR1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0800C + n*0x0040 + CCi	TOM Channel n CCU0 compare register (TOM[i]_CH[n]_CM0)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08010 + n*0x0040 + CCi	TOM Channel n CCU1 compare register (TOM[i]_CH[n]_CM1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08014 + n*0x0040 + CCi	TOM Channel n CCU0 counter register (TOM[i]_CH[n]_CNO)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08018 + n*0x0040 + CCi	TOM channel n status (TOM[i]_CH[n]_STAT)	R	0x0000_0001	See GTM Spec. ⁽²⁾
0x0801C + n*0x0040 + CCi	TOM channel n interrupt notification register (TOM[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08020 + n*0x0040 + CCi	TOM channel n interrupt enable register (TOM[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08024 + n*0x0040 + CCi	TOM channel n software interrupt generation (TOM[i]_CH[n]_IRQ_FORCINT)		0x0000_0000	See GTM Spec. ⁽²⁾
0x08028 + n*0x0040 + CCi	Channel n IRQ mode configuration register (TOM[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0802C + n*0x0040 + CCi	Reserved			
0x08030 + CCi	TGC0 global control reg (TOM[i]_TGC0_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08034 + CCi	TGC0 action time base register (TOM[i]_TGC0_ACT_TB)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08038 + CCi	TGC0 force update control reg (TOM[i]_TGC0_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0803C + CCi	TGC0 internal trigger control (TOM[i]_TGC0_INT_TRIG)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x08070 + CCi	TGC0 enable/disable control reg (TOM[i]_TGC0_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08074 + CCi	TGC0 enable/disable status reg (TOM[i]_TGC0_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08078 + CCi	TGC0 output enable control reg (TOM[i]_TGC0_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0807C + CCi	TGC0 output enable status reg (TOM[i]_TGC0_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
x08230 + CCi	TGC1 global control register (TOM[i]_TGC1_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08234 + CCi	TGC1 action time base register (TOM[i]_TGC1_ACT_TB)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08238 + CCi	TGC1 force update control reg (TOM[i]_TGC1_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0823C + CCi	TGC1 internal trigger control (TOM[i]_TGC1_INT_TRIG)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08270 + CCi	TGC1 enable/disable control reg (TOM[i]_TGC1_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08274 + CCi	TGC1 enable/disable status reg (TOM[i]_TGC1_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x08278 + CCi	TGC1 output enable control reg (TOM[i]_TGC1_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0827C + CCi	TGC1 output enable status reg (TOM[i]_TGC1_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x08030 – 0x0803C) + m*0x0040 + CCi	Reserved (m= 2..7, 10..15)			
(0x08400 – 0x087FC) + CCi	Reserved			
0x09800 – 0x0ABFC	Reserved for TOM3 to TOM5			
0x0AC00 – 0x0CFFC	Reserved			
ARU-connected Timer Output Module (ATOM) Registers (DDi = i*0x800, i = 0..4, m = 0..7)				
0x0D000 + m*0x0080 + DDi	ATOM Channel m ARU read address register (ATOM[i]_CH[m]_RDADDR)	R/W	0x01FE_01FE	See GTM Spec. ⁽²⁾
0x0D004 + m*0x0080 + DDi	ATOM Channel m control register (ATOM[i]_CH[m]_CTRL)	R/W	0x0000_X000	See GTM Spec. ⁽²⁾
0x0D008 + m*0x0080 + DDi	ATOM Channel m CCU0 compare shadow register (ATOM[i]_CH[m]_SR0)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D00C + m*0x0080 + DDi	ATOM Channel m CCU1 compare shadow register (ATOM[i]_CH[m]_SR1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x0D010 + m*0x0080 + DDi	ATOM Channel m CCU0 compare register (ATOM[i]_CH[m]_CM0)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D014 + m*0x0080 + DDi	ATOM Channel m CCU1 compare register (ATOM[i]_CH[m]_CM1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D018 + m*0x0080 + DDi	ATOM Channel m CCU0 counter register (ATOM[i]_CH[m]_CN0)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D01C + m*0x0080 + DDi	ATOM Channel m status register (ATOM[i]_CH[m]_STAT)	R/W	0x0000_0001	See GTM Spec. ⁽²⁾
0x0D020 + m*0x0080 + DDi	ATOM channel m interrupt notification register (ATOM[i]_CH[m]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D024 + m*0x0080 + DDi	ATOM channel m interrupt enable register (ATOM[i]_CH[m]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D028 + m*0x0080 + DDi	ATOM channel m software interrupt generation (ATOM[i]_CH[m]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D02C + m*0x0080 + DDi	Channel m IRQ mode configuration register (ATOM[i]_CH[m]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x0D030 – 0x0D03C) + m*0x0080 + DDi	Reserved for ATOM[i]_CH[m]			
0x0D040 + DDi	AGC Global control register (ATOM[i]_AGC_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D044 + DDi	AGC0 Enable/disable control register (ATOM[i]_AGC_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D048 + DDi	AGC Enable/disable status register (ATOM[i]_AGC_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D04C + DDi	AGC Action time base register (ATOM[i]_AGC_ACT_TB)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D050 + DDi	AGC Output enable control register (ATOM[i]_AGC_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D054 + DDi	AGC Output enable status register (ATOM[i]_AGC_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D058 + DDi	AGC Force update control register (ATOM[i]_AGC_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x0D05C + DDi	AGC Internal trigger control register (ATOM[i]_AGC_INT_TRIG)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x0D040 – 0x0D05C) + m*0x0080 + DDi	Reserved for ATOM[i]_CH[m], m>0			
(0x0D060 – 0x0D07C) + m*0x0080 + DDi	Reserved for ATOM[i]_CH[m]			

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
(0x0D400 – 0x0D7FC) + DDi	Reserved for ATOM[i]			
0x0F800 – 0x12C40	Reserved for ATOM5 to ATOM11			
0x12C44 – 0x17FFC	Reserved			
F2A[i] Configuration Registers (m = 0...7, i = 0..1, EEi = 0x4000)				
0x18000+m*0x04 + EEi	F2A read channel address register (F2A[i]_CH[m]_ARU_RD_FIFO)	R/W	0x0000_01FE	See GTM Spec. ⁽²⁾
0x18020+m*0x04 + EEi	F2A stream x configuration register (F2A[i]_CH[m]_STR_CFG)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x18040 + EEi	F2A stream activation register (F2A[i]_ENABLE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x18044 - 0x1807C) + EEi	Reserved			
AFD[i] Registers (m = 0...7, i = 0..1, EEi = 0x4000)				
0x18080+m*0x10 + EEi	AFD FIFO m buffer access register (AFD[i]_CH[m]_BUFF_ACC)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x18084+m*0x10 - 0x1808C+m*0x10) + EEi	Reserved for AFD[i]			
(0x18100 - 0x183FC) + EEi	Reserved			
FIFO[i] Configuration Registers (m = 0...7, i = 0, EEi = 0x4000)				
0x18400+m*0x40 + EEi	FIFO Channel m control register (FIFO[i]_CH[m]_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x18404+m*0x40 + EEi	FIFO Channel m end address register (FIFO[i]_CH[m]_END_ADDR)	R/W	0x0000_0XXX ⁽³⁾	See GTM Spec. ⁽²⁾
0x18408+m*0x40 + EEi	FIFO Channel m start address register (FIFO[i]_CH[m]_START_ADDR)	R/W	0x0000_0XXX ⁽³⁾	See GTM Spec. ⁽²⁾
0x1840C+m*0x40 + EEi	FIFO Channel m upper watermark register (FIFO[i]_CH[m]_UPPER_WM)	R/W	0x0000_0060	See GTM Spec. ⁽²⁾
0x18410+m*0x40 + EEi	FIFO Channel m lower watermark register (FIFO[i]_CH[m]_LOWER_WM)	R/W	0x0000_0020	See GTM Spec. ⁽²⁾
0x18414+m*0x40 + EEi	FIFO Channel x status register (FIFO[i]_CH[m]_STATUS)	R	0x0000_0005	See GTM Spec. ⁽²⁾
0x18418+m*0x40 + EEi	FIFO Channel m fill level register (FIFO[i]_CH[m]_FILL_LEVEL)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x1841C+m*0x40 + EEi	FIFO Channel m write pointer register (FIFO[i]_CH[m]_WR_PTR)	R	0x0000_0XXX ⁽³⁾	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x18420+m*0x40 + EEi	FIFO Channel m read pointer register (FIFO[i]_CH[m]_RD_PTR)	R	0x0000_0XXX ⁽³⁾	See GTM Spec. ⁽²⁾
0x18424+m*0x40 + EEi	FIFO m interrupt notification register (FIFO[i]_CH[m]_IRQ_NOTIFY)	R/W	0x0000_0005	See GTM Spec. ⁽²⁾
0x18428+m*0x40 + EEi	FIFO m interrupt enable register (FIFO[i]_CH[m]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x1842C+m*0x40 + EEi	FIFO m register to force interrupt by software (FIFO[i]_CH[m]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x18430+m*0x40 + EEi	FIFO m IRQ mode control register (FIFO[i]_CH[m]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x18434+m*0x40 + EEi	FIFO m error interrupt enable register (FIFO[i]_CH[m]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x18438+m*0x40 – 0x1843C+m*0x40) + EEi	Reserved			
(0x18600 – 0x18FFC) + EEi	Reserved for FIFO[i]			
(0x19000 – 0x19FFC) + EEi	FIFO[i] Memory			See GTM Spec. ⁽²⁾
(0x1A000 – 0x1BFFC) + EEi	Reserved			
0x1C000 – 0x1DFFC	Reserved for F2A1, AFD1, FIFO1, and FIFO1 MEM			
0x1E000 – 0x27FFc	Reserved			
Digital PLL Module (DPLL)				
0x28000	Control Register 0 (DPLL_CTRL_0)	R/W	0x003B_BA57	See GTM Spec. ⁽²⁾
0x28004	Control Register 1 (DPLL_CTRL_1)	R/W	0xB000_0000	See GTM Spec. ⁽²⁾
0x28008	Control Register 2 (actions 0–7 enable) (DPLL_CTRL_2)	R/W	x0000_0000	See GTM Spec. ⁽²⁾
0x2800C	Control Register 3 (actions 8–15 enable) (DPLL_CTRL_3)	R/W	x0000_0000	See GTM Spec. ⁽²⁾
0x28010	Control Register 4 (actions 16–24 enable) (DPLL_CTRL_4)	R/W	x0000_0000	See GTM Spec. ⁽²⁾
0x28014	Control Register 5 (actions 16–24 enable) (DPLL_CTRL_5)	R/W	x0000_0000	See GTM Spec. ⁽²⁾
0x28018	ACTION Status Register with connected shadow register (DPLL_ACT_STA)	R/W	x0000_0000	See GTM Spec. ⁽²⁾
0x2801C	Offset and switch old/new address register (DPLL_OSW)	R/W	0x0000_0200	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x28020	Address offset register for APT in RAM region 2 (DPLL_AOSV_2)	R/W	0x1810_0800	See GTM Spec. ⁽²⁾
0x28024	Actual RAM pointer to RAM regions 2a, b and d (DPLL_APT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28028	Actual RAM pointer to regions 1c1, 1c2 and 1c4 (DPLL_APS)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2802C	Actual RAM pointer to RAM region 2c (DPLL_APT_2c)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28030	Actual RAM pointer to RAM region 1c3 (DPLL_APS_1c3)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28034	Number of recent TRIGGER events used for calculations (mod 2*(TNU +1-SYN_NT)) (DPLL_NUTC)	R/W	0x0001_2001	See GTM Spec. ⁽²⁾
0x28038	Number of recent STATE events used for calculations (mod 2*(SNU +1-SYN_NS) (DPLL_NUSC)	R/W	0x0000_2081	See GTM Spec. ⁽²⁾
0x2803C	Number of active TRIGGER events to interrupt (DPLL_NTI_CNT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28040	Interrupt notification register (DPLL_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28044	Interrupt enable register (DPLL_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28048	Interrupt force register (DPLL_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2804C	Interrupt mode register (DPLL_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28050	Error Interrupt enable register (DPLL_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28054 – 0x280AC	Reserved			
0x280B0	Counter for pulses for TBU_TS1 to be sent in automatic end mode (DPLL_INC_CNT1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x280B4	Counter for pulses for TBU_TS2 to be sent in automatic end mode when SMC = RMO = 1w (DPLL_INC_CNT2)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x280B8	Old RAM pointer and offset value for TRIGGER (DPLL_APT_SYNC)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x280BC	Old RAM pointer and offset value for STATE (DPLL_APS_SYNC)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x280C0	TBU_TS0 value at last TRIGGER event (DPLL_TBU_TS0_T)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x280C4	TBU_TS0 value at last STATE event (DPLL_TBU_TS0_S)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x280C8	Direct load input value for SUB_INC1 (DPLL_ADD_IN_LD1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x280CC	Direct load input value for SUB_INC2 (DPLL_ADD_IN_LD2)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x280D0 – 0x280F8	Reserved for DPLL			
0x280FC	Status Register (DPLL_STATUS)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28100 + n*0x0004	9 bit ID information for input signal PMTR_n (8:0), n = 0...23 (DPLL_ID_PMTR_n)	R/W	0x0000_01FE	See GTM Spec. ⁽²⁾
0x28160 – 0x281DC	Reserved for DPLL			
0x281E0	shadow register of DPLL_CTRL_0 (DPLL_CTRL_0_SHADOW_TRIGGER)	R/W	0x0000_0257	See GTM Spec. ⁽²⁾
0x281E4	shadow register of DPLL_CTRL_0 (DPLL_CTRL_0_SHADOW_STATE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x281E8	shadow register of DPLL_CTRL_1 (DPLL_CTRL_1_SHADOW_TRIGGER)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x281EC	shadow register of DPLL_CTRL_1 (DPLL_CTRL_1_SHADOW_STATE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x281F0 – 0x281F8	Reserved			
0x281FC	Initialization Control and Status for RAMs (DPLL_RAM_INI)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
DPLL RAM 1A Map				
	RAM Region 1a (0.288 KB for 96 words of 24 bits)			
0x28200 + n*0x0004	ACTION_n Position/Value action request Register (PSAn), n = 0...23	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28260 + m*0x0004	ACTION_m time to react before PSAm (DLAm), m = 0...23	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x282C0 + n*0x0004	Number of TRIGGER/STATE increments to ACTION_n (NAn), n = 0...23	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28320 + m*0x0004	Calculated relative time to ACTION_m (DTAm), m = 0...23	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28380 – 0x283FC	Reserved	R/W		
DPLL RAM 1B Map (Regions 1b and 1c)				
	DPLL RAM Region 1b (0.375 KB for 128 words of 24 bits)			
	TRIGGER signal information stored			
	Note: The switch of the LSB address bits is performed using the SWON register at 0x0020.			
0x28400/ 0x0404	Actual signal TRIGGER time stamp register TRIGGER_TS (TS_T)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28404/ 0x0400	Previous signal TRIGGER time stamp register TRIGGER_TS_old (TS_T_old)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28408	Actual signal TRIGGER filter and signal value (FTV_T)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2840C	Reserved ⁽⁴⁾			

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
	STATE signal information stored Note: The switch of the LSB address bits is performed using the SWON register at 0x0020.	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28410/ 0x0414	Actual signal STATE time stamp register STATE_TS (TS_S)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28414/ 0x0410	Previous signal STATE time stamp register STATE_TS_old (TS_S_old)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28418	Actual signal STATE filter and signal value (FTV_S)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2841C	Reserved ⁽⁴⁾			
0x28420	TRIGGER hold time min value (THMI)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28424	TRIGGER hold time max value (THMA)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28428	Measured last pulse time from valid to invalid TRIGGER slope (THVAL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2842C	Reserved ⁽⁴⁾			
0x28430	Time out value of active TRIGGER slope (TOV)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28434	Time out value of active STATE slope (TOV_S)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28438	Calculated ADD_IN value for normal SUB_INC1 generation (ADD_IN_CAL1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2843C	Calculated ADD_IN value for normal SUB_INC2 generation (ADD_IN_CAL2)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28440	missing pulses to be added/subtracted directly to SUB_INC1 and INC_CNT1 once (MPVAL1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28444	missing pulses to be added/subtracted directly to SUB_INC2 and INC_CNT2 once (MPVAL2)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28448	target number of TRIGGER pulses (NMB_T_TAR)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2844C	target number of TRIGGER pulses (NMB_T_TAR_old)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28450	target number of STATE pulses (NMB_S_TAR)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28454	target number of STATE pulses (NMB_S_TAR_old)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28458 – 0x2845C	Reserved ⁽⁴⁾			
0x28460	reciprocal value of expected increment duration (T) (RCDT_TX)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x28464	reciprocal value of expected increment duration (S) (RCDT_SX)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28468	reciprocal value of the expected nominal increment duration (T) (RCDT_TX_nom)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2846C	reciprocal value of the expected nominal increment duration (S) (RCDT_SX_nom)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28470	actual reciprocal value of TRIGGER (RDT_T_actual)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28474	actual reciprocal value of STATE (RDT_S_actual)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28478	Duration of last TRIGGER increment (DT_T_actual)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x2847C	Duration of last STATE increment (DT_S_actual)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28480	Absolute error of prediction for last TRIGGER increment (EDT_T)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x28484	Average absolute error of prediction up to the last TRIGGER increment (MEDT_T)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x28488	absolute error of prediction for last STATE increment (EDT_S)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x2848C	Average absolute error of prediction up to the last STATE increment (MEDT_S)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x28490	Expected duration of current TRIGGER increment (CDT_TX)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x28494	Expected duration of current STATE increment (CDT_SX)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x28498	Expected nominal duration of current TRIGGER increment (without consideration of missing events) (CDT_TX_nom)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x2849C	Expected nominal duration of current STATE increment (without consideration of missing events) (CDT_SX_nom)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x284A0	TRIGGER Locking Range (TLR)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x284A4	STATE Locking Range (SLR)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28500 + n*0x0004	For ACTION_n in normal mode (PDT_n), n = 0...31	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x028580 – 0x285BC	Reserved ⁽⁴⁾			
0x285C0	Calculated number of sub-pulses between two STATE events (to be set by CPU) (MLS1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x285C4	Calculated number of sub-pulses between two STATE events (to be set by CPU) for the use when SMC = RMO = 1B (MLS2)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285C8	number of sub-pulses of SUB_INC1 in continuous mode, updated by the host only (CNT_NUM1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285CC	number of sub-pulses of SUB_INC2 in continuous mode, updated by the host only (CNT_NUM2)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285D0	Plausibility value of next active TRIGGER slope (PVT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285E0	Accurate calculated position stamp of last TRIGGER input (PSTC)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285E4	Accurate calculated position stamp of last STATE input (PSSC)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285E8/ 0x285EC	Measured position stamp of last TRIGGER input (PSTM)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285EC/ 0x285E8	Measured position stamp of last but one TRIGGER input (PSTM_old)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285F0/ 0x285F4	Measured position stamp of last STATE input (PSSM)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285F4/ 0x285F0	Measured position stamp of last but one STATE input (PSSM_old)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285F8	Number of pulses of current increment in normal mode for SUB_INC1 (NMB_T)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x285FC	Number of pulses of current increment in emergency mode for SUB_INC1 (NMB_S)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
	DPLL RAM Region 1c (0.75 KB for 256 words of 24 bits) Note: The following registers for the signal STATE are located in RAM Region 1c, read access by AEI via bus interface possible, writing results in an interrupt; data width of 3 bytes used for 24-bit values.			
	1c1			
0x28600 + n*0x0004	Nominal STATE reciprocal values in FULL_SCALE (RDT_Sn), n = 0...63 (2*(SNU+1)-SYN_NS) valid entries)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
	1c2			
0x28700 + m*0x0004	Time stamp field of STATE Events (TSF_Sm), m = 0...63 (2*(SNU+1) valid entries)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
	1c3			

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
0x28800 + n*0x0004	Adapt Values for all STATE Increments (ADT_Sn), n = 0...63 (2*(SNU+1-SYN_NS) valid entries)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
	1c4			
0x28900 + m*0x0004	Nominal STATE Increment Values for FULL_SCALE (DT_Sm), m = 0...63 (2*(SNU+1-SYN_NS) valid entries)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x28A00 – 0x28DFC	Reserved			
0x28E00 + n*0x0004	Calculated ACTION_n TIME STAMP (DPLL_TSACn), n = 0...31	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28E80 + n*0x0004	Calculated position value for action n (DPLL_PSACn), n = 0...31	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F00	control bits for actions 0...3 (DPLL_ACB_0)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F04	control bits for actions 4...7 (DPLL_ACB_1)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F08	control bits for actions 8...11 (DPLL_ACB_2)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F0C	control bits for actions 12...15 (DPLL_ACB_3)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F10	control bits for actions 16...19 (DPLL_ACB_4)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F14	control bits for actions 20...23 (DPLL_ACB_5)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F18	control bits for actions 24...27 (DPLL_ACB_6)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F1C	control bits for actions 28...31 (DPLL_ACB_7)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x28F20 – 0x2BFFC	Reserved			
DPLL RAM 2 map				
	Region 2a			
0x2C000 + n*0x0004	TRIGGER nominal increment reciprocals in FULL_SCALE (RDT_Tn), n = 0 ... N-1, with N = 2*(TNU+1-SYN_NT) valid entries, and N = 128...1024 OBS: final address 2a = 0x2C000 + (N-1)*0x4	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
	Region 2b			
final address 2a + 0x4 + n*0x0004	Time Stamp Field of TRIGGER events (TSF_Tn), n = 0...N-1, with N = 2*(TNU+1) valid entries, and N = 128...1024 OBS: the final address 2b depends on N	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
	Region 2c			

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
final address 2b + 0x4 + m*0x0004	Adapt Values for all Increments (ADT_Tm), m = 0...M-1, with M = 2*(TNU+1-SYN_NT) valid entries, and M = 128...1024 OBS: the final address 2c depends on M	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
Region 2d				
final address 2c + 0x4 + m*0x0004	nominal TRIGGER increment values for FULL_SCALE (DT_Tm), m = 0...M-1, with M = 2*(TNU+1-SYN_NT) valid entries, and M = 128...1024 OBS: the final address 2d depends on M	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
Multi Channel Sequencer (MCS) Configuration Registers (FFi = i*0x1000, i = 0..3, n = 0..7)				
0x30000+n*0x80 + y*0x04 + FFi	MCS Channel n GPR registers (MCS[i]_CH[n]_R[y]), y = 0...7	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x30020+n*0x80 + FFi	MCS Channel n control register (MCS[i]_CH[n]_CTRL)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x30024+n*0x80 + FFi	MCS Channel n ACB register (MCS[i]_CHn]_ACB)	R	0x0000_0000	See GTM Spec. ⁽²⁾
0x30028 + FFi	MCS Clear trigger control register (MCS[i]_CTRG)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x3002C + FFi	MCS Set trigger control register (MCS[i]_STRG)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x30028 – 0x3002C) + n*0x80 + FFi	Reserved (n > 0)			
(0x30030 – 0x3003C) + n*0x80 + FFi	Reserved			
0x30040+n*0x80 + FFi	MCS Channel n Program counter register (MCS[i]_CH[n]_PC)	R/W	0x0000_0000 + 4*n	See GTM Spec. ⁽²⁾
0x30044+x*0x80 + FFi	MCS Channel n interrupt notification register (MCS[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x30048+n*0x80 + FFi	MCS Channel n interrupt enable register (MCS[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x3004C+n*0x80 + FFi	MCS Channel n software interrupt generation register (MCS[i]_CH[n]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x30050+n*0x80 + FFi	IRQ mode configuration register (MCS[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x30054+n*0x80 + FFi	MCS Channel n error interrupt enable register (MCS[i]_CH[n]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾

Table 2015. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value ⁽¹⁾	Location
(0x30058 – 0x30070) + n*0x80 + FFi	Reserved			
0x30074 + FFi	MCS Control register (MCS[i]_CTRL)	R/W	0x000X_0000	See GTM Spec. ⁽²⁾
0x30078 + FFi	MCS Channel reset register (MCS[i]_RST)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
0x3007C + FFi	MCS Error register (MCS[i]_ERR)	R/W	0x0000_0000	See GTM Spec. ⁽²⁾
(0x30074 – 0x3007C) + n*0x80 + FFi	Reserved (n > 0)			
(0x30400 – 0x30FFC) + FFi	Reserved			
0x34000 – 0x36FFC	Reserved for MCS4 / 5 / 6 registers			
0x37000 – 0x37FFC	Reserved			
MCS Memories (GGi = i*0x8000, i = 0..3)				
(0x38000 – 0x3BFFC) + GGi	MCS Memory (MCS[i]_MEM)			See GTM Spec. ⁽²⁾
(0x3C000 – 0x3FFFC) + GGi	Reserved			
0x58000 – 0x6BFFC	Reserved for MCS4 / 5 / 6 Memories			
0x6C000 – 0xFFFFC	Reserved			

1. Default value when in reset state.
2. For detailed information see the *GTM-IP Specification* document.
3. Reset value depends on configuration parameters.
4. RAM reserved position for GTM-IP internal use. No error occurs when accessed.

88.2.2 Register Descriptions

This section contains the description of additional registers on the top integration module. Description of GTM-IP registers, or GTMDI registers are found in their respective specification document.

88.2.2.1 GTM Module Configuration Register (GTMMCR)

The GTMMCR contains the control bits to configure the general operation of the GTMINT and GTM-IP Module.

Figure 2157. GTM Module Configuration Register (GTMMCR)

Offset 0x000C0

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MDIS	0	0	0	0	0	0	0	0	0	0	0	0	0	AEISREN
W																
Reset	0	— ⁽¹⁾	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	STPS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	— ⁽¹⁾	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset value is SoC dependent.

GTMMCR register functions are as shown in [Table 2016](#).

Table 2016. GTMMCR field descriptions

Field	Description
1 MDIS	MDIS Module Disable. The MDIS bit puts the GTMINT Module in stop/disabled mode. Communication through the slave-bus Interface is ignored in this mode except reads/writes to the GTMMCR register and reads from other integration registers which are allowed. 1 Module disable request 0 Normal Mode
15 AEISREN	AEISREN AEI interface soft-reset control enable. The AEISREN bit enables the soft-reset of the AEI interface that is controlled by AEISRST bit on GTMMAEICR register.. Please, refer to Section 88.3.7, AEI Interface Software Reset Description for more details. 1AEI soft-reset control is enabled 0AEI soft-reset control is disabled
17 STPS	STPS Stop Mode Status. The STPS bit indicates the GTM-IP Module is in stop mode, or by MDIS bit field setting or by stop line request. Please, refer to Section 88.3.4, Stop Mode Description for more details. 1GTM is in Stop Mode 0Normal Mode

88.2.2.2 GTM Interrupts Clear Register (GTMINTCLR)

The GTMINTCLR contains the control bits to clear some specific interrupt request output.

Figure 2158. GTM Interrupts Clear Register (GTMINTCLR)

Offset 0x000C8												Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
W							INTCLR_PTR[0:9]													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

GTMINTCLR register functions are as shown in [Table 2017](#).

Table 2017. GTMINTCLR field descriptions

Field	Description
22-31 INTCLR_PTR[0:9]	INTCLR_PTR Interrupt Clear Pointer. This field value indicates / selects which interrupt will be cleared. The correspondence between the number and the interrupt is given in Table 2018 . Please, refer to Section 88.3.8, Interrupt and DMA Support Logic for more details.

Table 2018. Correspondence between the INTCLR_PTR and the interrupts outputs

INTCLR_PTR[0:9]		Interrupt Output Signal to be cleared
HEX	DEC	
0x000	0	No interrupt is selected
0x001 to 0x2C1	1 to 705	Reserved
0x2C2	706	ipi_int_gtm_aei
0x2C3 to 0x2C5	707 to 709	ipi_int_gtm_aru[0] - [2]
0x2C6	710	ipi_int_gtm_brc
0x2C7	711	ipi_int_gtm_cmp
0x2C8	712	ipi_int_gtm_spe0
0x2C9	713	ipi_int_gtm_spe1
0x2CA to 0x2D1	714 to 721	ipi_int_gtm_psm0[0] - [7]

Table 2018. Correspondence between the INTCLR_PTR and the interrupts outputs

INTCLR_PTR[0:9]		Interrupt Output Signal to be cleared
HEX	DEC	
0x2D2 to 0x2EC	722 to 748	ipi_int_gtm_dp1l[0] - [26]
0x2ED to 0x2F4	749 to 756	ipi_int_gtm_tim0[0] - [7]
0x2F5 to 0x2FC	757 to 764	ipi_int_gtm_tim1[0] - [7]
0x2FD to 0x304	765 to 772	ipi_int_gtm_tim2[0] - [7]
0x305 to 0x30C	773 to 780	ipi_int_gtm_tim3[0] - [7]
0x30D to 0x314	781 to 788	ipi_int_gtm_mcs0[0] - [7]
0x315 to 0x31C	789 to 796	ipi_int_gtm_mcs1[0] - [7]
0x31D to 0x324	797 to 804	ipi_int_gtm_mcs2[0] - [7]
0x325 to 0x32C	805 to 812	ipi_int_gtm_mcs3[0] - [7]
0x32D to 0x334	813 to 820	ipi_int_gtm_tom0[0] - [7]
0x335 to 0x33C	821 to 828	ipi_int_gtm_tom1[0] - [7]
0x33D to 0x344	829 to 836	ipi_int_gtm_tom2[0] - [7]
0x345 to 0x348	837 to 840	ipi_int_gtm_atom0[0] - [3]
0x349 to 0x34C	841 to 844	ipi_int_gtm_atom1[0] - [3]
0x34D to 0x350	845 to 848	ipi_int_gtm_atom2[0] - [3]
0x351 to 0x354	849 to 852	ipi_int_gtm_atom3[0] - [3]
0x355 to 0x358	853 to 856	ipi_int_gtm_atom4[0] - [3]
0x359 to 0x3FF	857 to 1023	Reserved

88.2.2.3 GTM AEI Control Register (GTMMAEICR)

The GTMMAEICR contains the control bits to the AEI interface in GTM-IP.

Figure 2159. GTM AEI Control Register (GTMMAEICR)

Offset 0x000CC												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																AEISRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GTMMAEICR register functions are as shown in [Table 2019](#).

Table 2019. GTMMAEICR field descriptions

Field	Description
31 AEISRST	AEISRST AEI interface soft-reset control. The AEISRST bit when set applies a reset on the GTM-IP AEI interface. The set is enabled by the AEISREN bit on GTMMCR register. This is a self clear bit and is always read as 0.. Please, refer to Section 88.3.7, AEI Interface Software Reset Description for more details.

88.3 Functional Description

The following sections describe the GTMINT Module functionality.

The GTMINT Module is designed to accommodate the multipurpose timer (GTM-IP) and the debugger block (GTMDI). The GTM-IP can generate timed signals and also can handle external signals to obtain their timing information. The GTMDI provides several features to help the development of programs to be run on GTM-IP. Please refer to [Figure 2156](#) for the integration architecture.

The GTMINT Module integrates the following internal sub-modules into the SPC57EM80xx platform:

- GTM-IP timer
- GTMDI debug interface
- ECC error correction sub-blocks
- Interface match logic

The following sections describe each one of these sub-modules and their combined operation.

88.3.1 GTM-IP Timer

The Generic Timer Module (GTM) is composed of several submodules of different functionality. These submodules can be combined in a configurable manner to form a complex timer module that serves different application domains and different classes within one application domain. Because of this scalability and configurability the timer is called generic. The scalability and configurability is reached with an architecture philosophy where dedicated hardware submodules are located around a central routing unit called Advanced Routing Unit (ARU). The ARU can connect the submodules in a flexible manner. The connectivity is software programmable and can be configured during runtime.

Nevertheless, the GTM-IP is designed to unload the CPU or a peripheral core from a high interrupt load. Most of the tasks inside the GTM-IP can run (once set up by an external CPU) independent and in parallel to the software. There may be special situations, where the CPU has to take action but the goal of the GTM design was to reduce these situations to a minimum.. In addition, the timer section of GTM-IP runs at a fixed clock frequency (80MHz) to allow real time period counting, but the interface to CPU can work with a frequency variable clock.

88.3.1.1 GTM-IP Timer Architecture

The hardware submodules have dedicated functions, e.g., there are timer input modules where incoming signals can be captured and characterized together with a notion of time. By combination of several submodules through the ARU complex functions can be established. For example, the signals characterized at an input module can be routed to a signal processing unit where an intermediate value about the incoming signal frequency can be calculated.

The modules that help to implement such complex functions are called infrastructural components further on. These components are present in all GTM variants. However, the number of these components may vary from device to device. Other submodules have a more general architecture and can fulfill typical timer functions, e.g., there are PWM generation units. The third class of submodules are those fulfilling a dedicated functionality for a certain application domain, e.g., the DPLL serves engine management applications. A fourth group of submodules is responsible for supporting the implementation of safety functions to fulfill a defined safety level.

Each GTM-IP is build up therefore with submodules coming from those four groups. The application class is defined by the amount of components of those submodules integrated into the implemented GTM-IP.

A list of submodules is given in [Table 2020](#) below.

Table 2020. Submodules of GTM-IP

Item	Submodule	Group
1	Advanced Routing Unit (ARU)	Infrastructural components
2	Broadcast Module (BRC)	
3	First In First Out Module (FIFO)	
4	AEI-to-FIFO Data Interface (AFD)	
5	FIFO-to-ARU Interface (F2A)	
6	Clock Management Unit (CMU)	
7	Time Base Unit (TBU)	
8	Timer Input Module (TIM)	IO Modules
9	Timer Output Module (TOM)	
10	ARU-connected Timer Output Module (ATOM)	
11	Multi Channel Sequencer (MCS)	Signal generation and processing
12	Memory Configuration Module (MCFG)	Infrastructural component for MCS
13	TIM0 Input Mapping Module (MAP)	Dedicated
14	Digital PLL (DPLL)	
15	Sensor Pattern Evaluation Module (SPE)	BLDC support
16	Interrupt Concentrator Module (ICM)	Interrupt services
17	Output Compare Unit (CMP)	Safety features
18	Monitoring Unit (MON)	

For more details, please refer to the *GTM-IP Specification* document.

88.3.2 GTMDI Debug Interface

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including MCS cores, ARU data trace, Input Channel Filters trace, Output Timer channels, DPLL and SPE sub-modules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules in order to implement complex debug features. Trace capability is also provided along with a global time-stamp which allows real time trace for several GTM sub-module events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and Nexus Aurora router on-chip modules. Please, note that the Nexus standard is only used for the physical interfaces. In order to implement the debug features such as breakpoint and messages required for the GTM debug, the Nexus standard could not be followed.

For detailed architecture and features description, please refer to the GTMDI chapter of this reference manual.

88.3.2.1 Peripheral Slave Bus Read Debug Access

Debugger is able to read all GTM registers, which are visible from the peripheral bus even when the timer is running. Read-access of the debugger is possible without destroying or

changing the contents of these registers. A read debug access using the slave bus has no side effects on the timer. The main example is a read access to the FIFO that does not update the FIFO pointers.

88.3.3 Error Detection on GTM Memories Accesses

The GTM-IP architecture requires 12 RAM units that should be placed on the SoC. Each RAM is connected with separated input and output ports to the GTM-IP for safety purposes.

Memories of this block are equipped with error detection features for safety improvement. The following procedures detect errors on memory accesses:

- Memory write data and its address are considered in the parity bits calculation. These parity bits are stored in that memory address as part of the written data.
- Memory read data and its address are considered in the error detection. The parity bits are part of the data read from the specified memory address.
- Error Correcting Code (ECC), that corrects single errors and detects double errors on each read access

88.3.3.1 ECC Description

The Error Correcting Code (ECC) for GTM RAMs provides guaranteed single bit error correction and double bit error detection (without correction) per accessed word. On write accesses, the parity bits are calculated and appended to the original data bits to be written into the specified memory address. And error detection and correction are performed on read accesses.

Note: In case of error detection (with correction or not), no write operation is done here to fix possible errors in the memory content.

Note: All memories must be initialized with write operations in all addresses because the ECC parity bits need to be stored in each memory address before any read operation.

GTM-IP sub-block does this initialization automatically for all memories, except for FIFO0 RAM. In this case, the user must be careful to execute writes before reads.

[Table 2021](#) presents a summary of GTM memories and the necessary ECCs for single error correction / double error detection.

Table 2021. ECC Characteristics for GTM Memories

Memory	MEM Number of Words	MEM Address Width (bits)	Uncoded Data Width (bits)	ECC (bits)	MEM Data In/Out (Data + ECC) (bits)	Matrix Size (N.o Synd.)
FIFO0 RAM	1024	10	29	7	36	46
DPLL RAM1A	96	7	24	7	31	38
DPLL RAM1B	384	9	24	7	31	40
DPLL RAM2	2048	11	24	7	31	42
MCS0 RAM0	1024	10	32	7	39	49
MCS1 RAM0	1024	10	32	7	39	49
MCS2 RAM0	1024	10	32	7	39	49

Table 2021. ECC Characteristics for GTM Memories

Memory	MEM Number of Words	MEM Address Width (bits)	Uncoded Data Width (bits)	ECC (bits)	MEM Data In/Out (Data + ECC) (bits)	Matrix Size (N.o Synd.)
MCS3 RAM0	1024	10	32	7	39	49
MCS0 RAM1	512	9	32	7	39	48
MCS1 RAM1	512	9	32	7	39	48
MCS2 RAM1	512	9	32	7	39	48
MCS3 RAM1	512	9	32	7	39	48

88.3.3.1.1 Error Correcting Code (ECC) and Syndrome Definition

This ECC is based on the Hsiao code with 7 parity bits for each data word.

[Table 2022](#) shows the definition of the syndrome field for each bit position. Note that the ECC data vector width can be different for each memory, so we consider only the necessary positions for each one, not all the table positions.

Table 2022. 42 bits Word Based Syndrome Definition

SYND[6:0] / Element Value (hex)	Error Result (for SMATRIX)	Matrix Element (for MATRIX)	SYND[6:0] / Element Value (hex)	Error Result (for SMATRIX)	Matrix Element (for MATRIX)
40	error on code bit 6	-	54	error on data bit 23	element 23
20	error on code bit 5	-	15	error on data bit 22	element 22
10	error on code bit 4	-	16	error on data bit 21	element 21
08	error on code bit 3	-	34	error on data bit 20	element 20
04	error on code bit 2	-	25	error on data bit 19	element 19
02	error on code bit 1	-	26	error on data bit 18	element 18
01	error on code bit 0	-	64	error on data bit 17	element 17
4F	error on data ⁽¹⁾ bit 41	element 41	2C	error on data bit 16	element 16
3E	error on data bit 40	element 40	1A	error on data bit 15	element 15
3D	error on data bit 39	element 39	23	error on data bit 14	element 14
3B	error on data bit 38	element 38	2A	error on data bit 13	element 13
37	error on data bit 37	element 37	32	error on data bit 12	element 12
2F	error on data bit 36	element 36	46	error on data bit 11	element 11
1F	error on data bit 35	element 35	4A	error on data bit 10	element 10
70	error on data bit 34	element 34	52	error on data bit 9	element 9
68	error on data bit 33	element 33	62	error on data bit 8	element 8
07	error on data bit 32	element 32	13	error on data bit 7	element 7

Table 2022. 42 bits Word Based Syndrome Definition

SYND[6:0] / Element Value (hex)	Error Result (for SMATRIX)	Matrix Element (for MATRIX)	SYND[6:0] / Element Value (hex)	Error Result (for SMATRIX)	Matrix Element (for MATRIX)
49	error on data bit 31	element 31	29	error on data bit 6	element 6
0D	error on data bit 30	element 30	31	error on data bit 5	element 5
0E	error on data bit 29	element 29	43	error on data bit 4	element 4
38	error on data bit 28	element 28	45	error on data bit 3	element 3
4C	error on data bit 27	element 27	19	error on data bit 2	element 2
1C	error on data bit 26	element 26	51	error on data bit 1	element 1
58	error on data bit 25	element 25	61	error on data bit 0	element 0
0B	error on data bit 24	element 24	-	-	-
-	-	-	-	-	-
00	no error	-	any other ⁽²⁾	non-correctable error	-

1. Actually the data can be data or address bit.
2. For the FIFO0 RAM with 29 data bits and 10 address bits, the valid error positions are from bit 0 to bit 38, only. The syndromes with values for bit 39 to 41 of the table are considered also for non-correctable errors.

88.3.3.2 Error Report Description

Each RAM generates a defined number of information to report an error on an access. Each of them is described below.

- Error Address - this is the mapped address of the RAM as defined in [Section 88.2.1, Module Memory Map](#). This field has only meaning when an error occurs and is flagged by the two error signals below. It is reported on read or write operations.
- Non-correctable Error - this one bit width error indicates that:
 - The ECC detected errors in the read access and it is not possible to correct them because the ECC can only fix single bit error. The error can occur in the RAM read data bus (parity bits + data) or in the address value that the RAM has received.
- Correctable Error - this one bit width error indicates that:
 - The ECC detected and corrected a single bit error on the RAM read data bus (parity bits + data). The parity bits are discarded after the ECC decoder but this error report includes these bits to obtain the performance of the whole RAM data storage sector..

This Error Report remains active for only one RAM clock cycle and should be captured by the external RAM Error Monitor module. These error indications are also available for the GTM-IP timer module that is the block that requests the RAM accesses. Uncorrectable ECC error indication are provided to GTM-IP but the reaction to the error depends on this block itself. Please refer to its specification for more details.

88.3.4 Stop Mode Description

The stop mode is characterized by the stopping of the functional clock and the timer configuration registers clock. Therefore, the timers become stopped and almost all registers access are not allowed. This mode can be requested by two ways:

- By asserting the bit field MDIS in the GTMMCR configuration register.
- By receiving the SoC stop request signal ipg_stop.
 - the minimum waiting time to a new stop request is about 20 bus clock cycles.

However, it is important to note that to confirm the stop request was granted, it is necessary to read the stop status bit field STPS in the GTMMCR register.

88.3.4.1 Stop Mode by MDIS

The stop mode is requested by asserting this bit field with a write operation on the GTMMCR mapped register. The module only requests to stop the clocks when no slave bus accesses are pending in the GTM-IP. When in stop mode:

- The clocks are disabled to feed the whole GTM-IP and peripheral functions like the ECC
- The stop status bit field STPS in GTMMCR register is set.
- Read or write accesses to all mapped registers are not allowed and generates an access error indication, except accesses for the GTMMCR register.
- The slave bus clock is not affected by stop mode.

This stop mode ends by clearing the MDIS bit field.. The STPS stop status bit is automatically cleared after a while (it is considered that the SoC stop request is not set).

88.3.5 Debug / Halt Mode Description

The debug / halt mode is requested via GTMDI and is used to debug the GTM-IP sub-block. In this mode it is required that the core of GTM-IP becomes halted by stopping its clock. Therefore all time bases and counters are frozen.

Debug or halt mode requests should be enabled by setting the bit field DBE on the GTMDI GTMDI_DC register that is accessible by the JTAG interface (refer to GTMDI block guide for more details). GTM halt is only activated when no read- or write-access to GTM over AEI are taking place in parallel. Only in this condition the GTM is released to change to debug / halt mode read.

It is important to notice that any peripheral slave bus access is active in this mode . Debugger is able to read all GTM addresses read-/writeable from the peripheral bus when GTM is running or halted. All registers in the register map can be accessed by the peripheral bus, even the RAMs because they are in the register map. Of course, the clocks for the RAMs are not stopped in debug mode.

For more details about the debug mode, please refer to GTMDI documentation.

Note: *If this module is in disable mode by setting the MDIS bit at the GTMMCR register, the debug request is not considered until the MDIS bit is cleared.*

88.3.6 GTM-IP Timer Software Reset Description

The GTM-IP timer portion has the software reset possibility by writing a 1 in the RST bit field on the GTM_RST register.. This bit is self cleared and is always read as zero. In addition, this bit is write protected by the bit field RF_PROT in the GTM_CTRL register.

This is a global reset to GTM-IP block only. The GTMDI is not affected by this soft-reset.

88.3.7 AEI Interface Software Reset Description

The GTM-IP has software reset control for the AEI interface sub-block only. This block is the slave bus interface of GTM-IP and is not reset by the GTM-IP timer soft-reset. The AEI soft-reset occurs by writing a 1 in the AEISRST bit field on the GTMMAEICR register. This bit is self cleared and is always read as zero. In addition, this bit is write protected by the bit field AEISREN in the GTMMCR register.

A soft-reset in the AEI interface is equivalent to the hard-reset. It occurs in the next clock cycle of the request and takes one bus clock cycle. The AEI buffer and configuration registers are cleared, and some slave bus access can be aborted. Therefore, access commands in the buffer memory are lost and the configuration registers of the AEI interface change to their reset values.

88.3.8 Interrupt and DMA Support Logic

The GTM-IP provides several interrupts / DMA request signals to signalize internal events to other external modules. The list of all signals are provided in the GTM-IP documentation and all have the same type of controls and configuration options.

Each interrupt / DMA request line represents a set of internal interrupt / DMA request signals, and each of this internal signal has its own set of status / control / configuration registers: IRQ_EN, IRQ_NOTIFY, IRQ_FORCINT, and IRQ_MODE registers. Please, refer to *GTM-IP specification* for more details.

It is required to maintain the configuration register IRQ_MODE=00 that is the Level Interrupt Mode, the default mode to be compatible with the chip standard. the IRQ_NOTIFY register is the internal flag and can be enabled to set an external signal by the IRQ_EN register. For more details, please refer to the GTM-IP documentation.

The GTM integration module provides the same GTM-IP set of signals as interrupt requests and as DMA requests and the real functionality on the chip can be configured later. However, if a signal is configured to start a DMA request, the request can be cleared by the respective DMA answer, but it can be also cleared by software as occurs for interrupt requests.

88.3.8.1 Interrupt Request Clear Controls

Each interrupt request output can be a composition of several internal events flags that are enabled by configuration registers. After an interrupt request is recognized by the chip interrupt controller and it is serviced, the interrupt routine must clear the flags that originated that service request. The GTM integration module provides two ways to clear those flags, but each method has its limitations that are described below.

The first way to clear the events flags that originated the interrupt request is by a slave bus write operation in the interrupt mapped registers named *_IRQ_NOTIFY in [Table 2015](#). The main characteristics are listed below:

- For example, writing a 1 in the bit field DEST_ERR on the register BRC_IRQ_NOTIFY is going to clear this bit to clear the ipi_int_gtm_brc interrupt output signal
- Several flags in this register could have generated the interrupt request, therefore all of them can be cleared with this write operation.
- There is a delay between the reception of the clear command and the actual clear of the flags. This time interval can vary and depends on the number of commands inside the AEI buffer. Therefore it is not possible to know easily when the flags were cleared.
- This method of interrupt clear can generate a new service request if we are using write masked response mode for AEI interface.

The second way to clear the events flags of the interrupt requests is by using the mapped register GTMINTCLR that is described in [Section 88.2.2.2, GTM Interrupts Clear Register \(GTMINTCLR\)](#). The main characteristics of this method are listed below:

- For example, to clear the ipi_int_gtm_brc interrupt output signal, we execute a write operation in the GTMINTCLR register with data value 0x05 in the INTCLR_PTR bit field. This value selects to execute a clear in all flags that are used to generate that interrupt request.
- There is a delay between the reception of the clear command and the actual clear of the flags. This time interval can vary and depends on the number of commands inside the AEI buffer. However, the flags clear instant is well defined and can be seen by the integration module.
- As this method clear all flags for an interrupt request output, some flag that is set during the interrupt service is also cleared, and then some event can be lost.

88.4 Initialization/Application Information

The GTM integration module receives two active low reset signals, one on the slave bus clock domain, and the other in the peripheral clock domain. The last one resets all the timer logic, and registers inside the GTM-IP block. Just after this reset signal is negated, the GTM-IP is in the following condition:

- The GTM bridge is in asynchronous mode.
- after reset, write operations waits the response status (write response is not masked).
- If MDIS_DEFAULT is 1, a write access to GTMMCR is necessary to clear the MDIS bit register and have the GTM-IP clocks toggling.
- DPLL RAMs and MCSs RAMs starts the write initialization cycles. This is necessary for ECC parity bits storage on each RAM address.
- FIFO0 RAM is not automatically initialized. In normal operation mode, the initialization is not required because the FIFO memory always have to written first to be read later.

Please, refer to GTM-IP and GTMDI documentation to initialize the GTMINT module.

89 Emulation and Debug Device Introduction

The emulation and debug device is a multi-die MCU that adds additional functionality specifically for ease in calibration and debug. The Emulation and debug device (ED) consists of the production die (PD) and a buddy die (BD).

89.1 Functionality

The ED is a PD packaged with a BD to provide additional capability during development. The ED is used for development, debug, and calibration of automotive powertrain controller applications for 4- to 8-cylinder engines. To enable these operations, the buddy device supports the following functions:

- Debug and calibration tool interface—Allows tool read/write access to buddy device resources even if main MCU supplies are removed as long as special ED debug supplies are maintained. The BD duplicates the DCI, JTAGM and LFAST functionality that is also available in the PD.
- High speed tool trace port (based on the IEEE-ISTO 5001-2011 Nexus Aurora standard)
- 1 MB on-chip SRAM usable as calibration overlay or trace RAM—entire SRAM contents are maintained as long as ED debug supply is maintained

89.2 Feature summary

On-chip modules available on the BD include the following features:

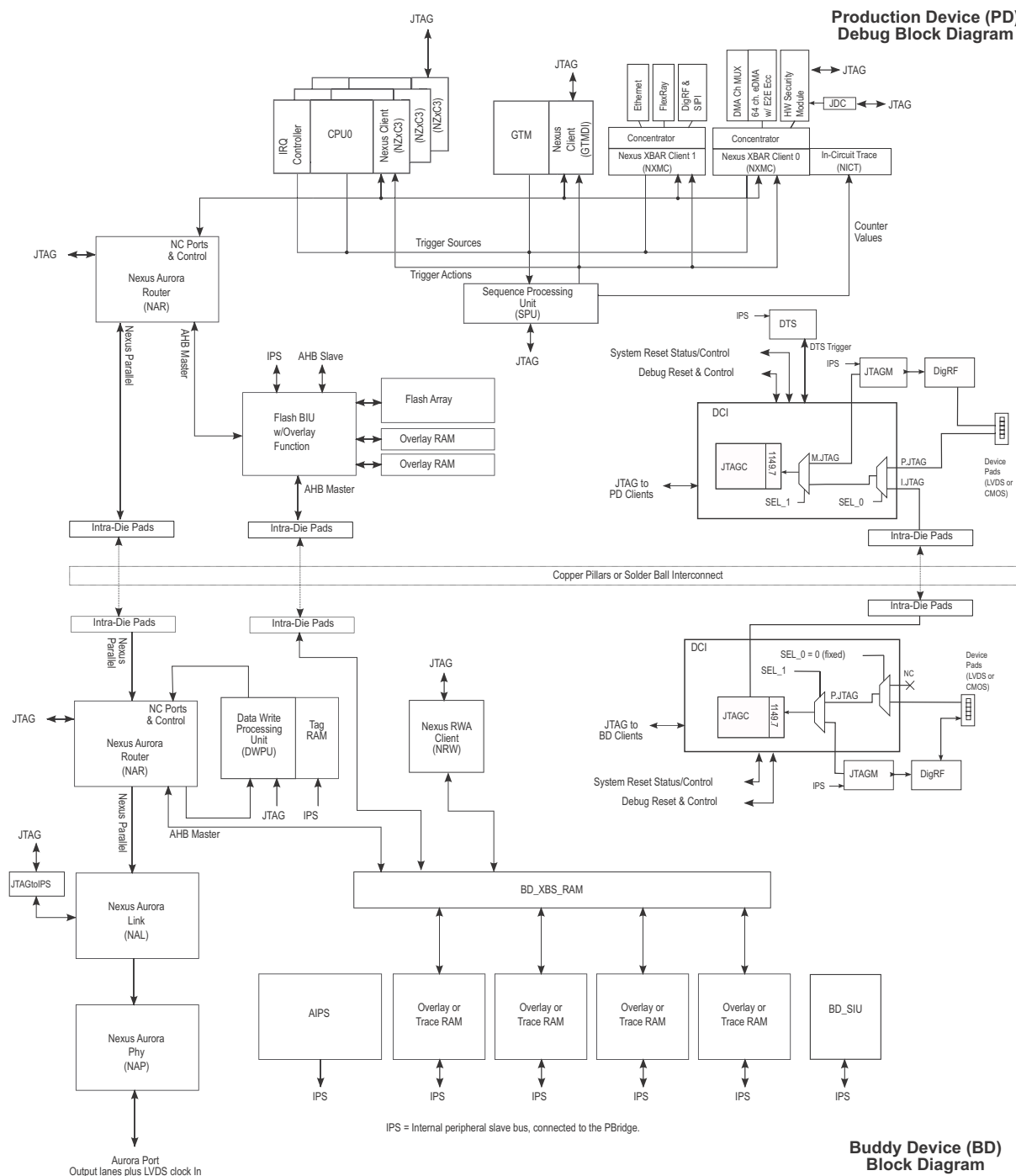
- 1 MB on-chip SRAM
 - Access timing matches internal PD flash accesses when used in conjunction with PD calibration remap logic
 - Includes ECC
 - Implemented as 4 × 256 KB partitions where each partition can be accessed by different masters without causing blockages to access of any other partition
- XBAR switch architecture
- XBAR slave interface
 - 64-bit data width with ECC checkbits
 - Supports “Early Write” AHB protocol (AHB-EW) for one clock trace data writes
 - Allows a PD with AHB master to support read and write to buddy device memory-mapped resources—Memory-mapped resources include buddy device SRAM and AIPS resources
- Aurora trace port
 - 1.25 Gbit/s maximum data rate per lane
 - Support for up to 8-lane LVDS data out
 - Four lanes supported in ED package

- Port configurable to operate in 2- or 4-lane width
 - One lane LVDS reference clock in
 - Supports Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2011 standard.
- Data Write Processing Unit (DWPU)
 - Filters data trace stream to retain only writes to user selected locations
 - Detects user configured out of range condition on writes to selected locations
- Debug and Calibration Interface—Supports both JTAG (1149.1 and 1149.7) and LFAST modes
- Independent high and low voltage supplies allow tool control of BD debug and calibration resources even when the main MCU supplies are off
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)
- Self test capability

89.3 Block diagram

This figure shows the top-level block diagram of the emulation and debug device.

Figure 2160. Emulation and debug device architecture



89.4 Clocking

Figure 2161 shows the clocking structure of the BD. There is one internal clock source and four external clock sources:

- Internal RC oscillator (IRC) clock
- System clock coming from the PD (identified as BD_CLK clock in the clock generation figure of the Clocking chapter)
- LFAST PLL reference clock coming from the PD (BD LFAST PLL clock in the clock generation figure of the Clocking chapter)
- JTAG clock coming from the TCK pin in the ED
- Aurora PHY clock coming from differential LVDS pins in the ED

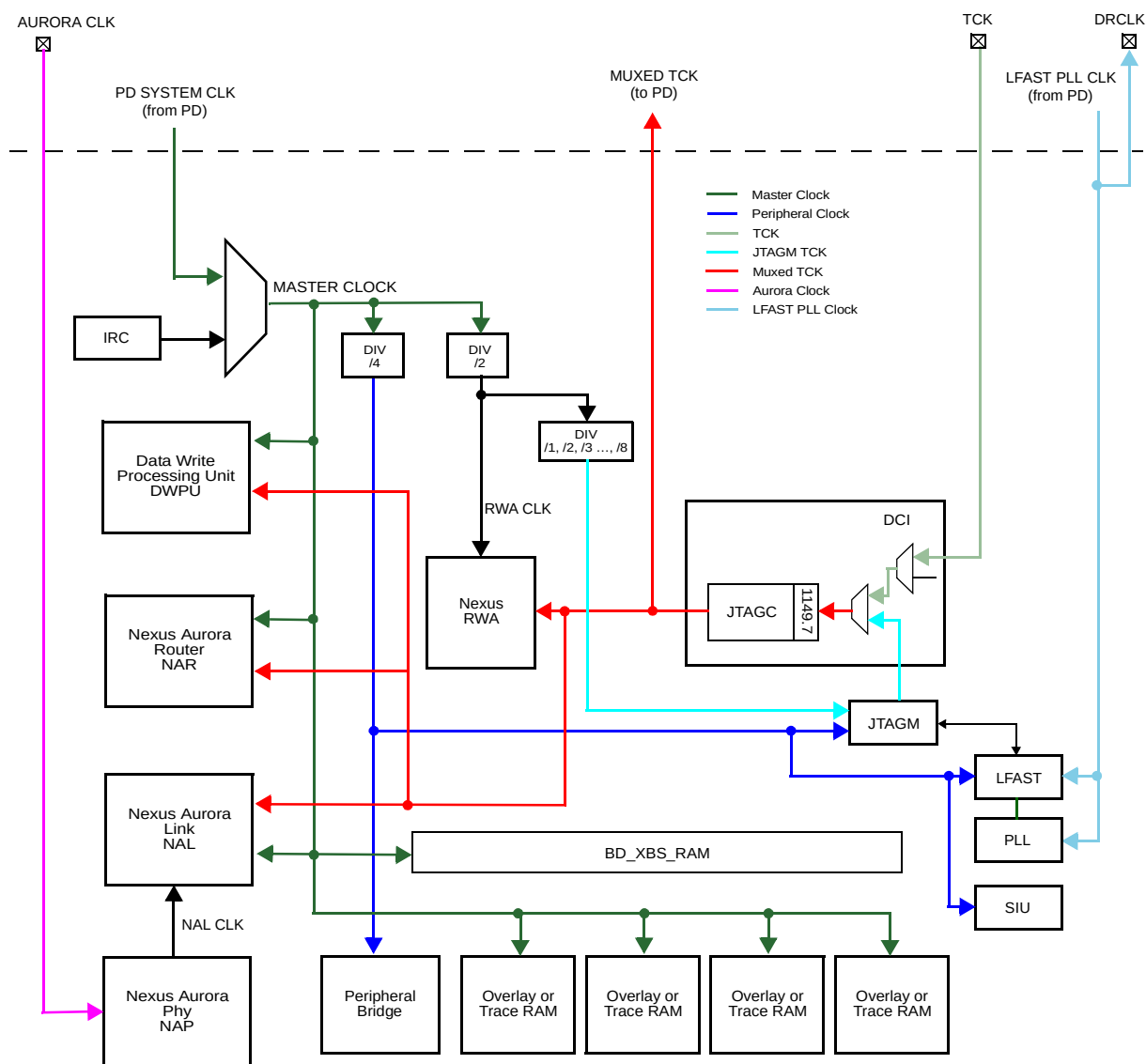


Figure 2161. BD clock structure

The IRC is the source of the BD master clock when the PD is unpowered, or powered but still in reset state. It has a nominal frequency of 64 MHz, does not have any user accessible register and is always enabled. It does not have any trimming capability, therefore the expected precision is on the order of $\pm 30\%$.

The PD system clock is the source of the BD master clock when the PD is powered and out of reset state. When the PD PLL is engaged and locked this clock has a nominal frequency of 200 MHz, but while the PLL is not engaged the frequency is that of the PD internal RC oscillator.

The master clock directly feeds the XBAR, SRAMs and the Aurora sub-system. It is also divided down to generate the following internal clocks:

- Peripheral clock (divided by 4)—provides the clock to memory mapped registers and other low-speed logic
- Nexus RWA clock (divided by 2)—provides the clock to the Nexus Read Write Access module
- JTAGM TCK (divided by 1, 2, ..., 8)—provides a high speed TCK clock to the JTAGM module

The LFAST PLL clock from the PD (also referred to as `lfast_sysclk`) feeds the LFAST PLL, but it can also directly clock the LFAST in the low data rate mode. This clock is also made available as an output clock to the DRCLK package pin.

The internal TCK clock (referred to as muxed clock in [Figure 2161](#)) is used in the Nexus RWA, NAR, NAL, and it also goes to the PD to be used as the JTAG clock. (In the ED, the JTAG package pins are bonded to the BD JTAG interface; therefore, the PD receives the JTAG signals from the BD.) This clock can come from two sources (see the Buddy Device–Debug and Calibration Interface (DCI) chapter):

- TCK package pin, when using 1149.1 or 1149.7 access
- Divided down version of the master clock to be used in LFAST/JTAGM debug accesses

Note: The division factor for the LFAST/JTAGM clock is defined by the `TCKSEL` field of the `JTAGM_MCR` (refer to the JTAGM chapter for more details). However the division factor described there applies to an already divided-by-2 clock, as shown in [Figure 2161](#).

When the PD is unpowered, access through LFAST/JTAGM is not possible. In this case, JTAG accesses can be done only by 1149.1 or 1149.7 modes accessing through the Nexus RWA module. In this mode, the TCK clock comes from the package pin and the RWA clock is a divided-by-2 version of the IRC; therefore, these clocks are asynchronous to each other. When clocks at the input of the RWA are asynchronous, there is a restriction that the RWA clock must be at least double the frequency of the JTAG clock. This means that the master clock frequency must be at least four times the JTAG clock frequency.

In the LFAST/JTAGM mode of operation, both RWA clock and internal TCK are synchronous to each other, therefore they can both be the same frequency. This means that TCK frequency can be half of the master clock frequency. [Table 2023](#) summarizes the frequency restrictions on the Nexus RWA accesses.

Table 2023. JTAG operating frequencies

PD configuration	BD configuration	PD system clock frequency [MHz]	BD master clock frequency [MHz]	Max RWA JTAG frequency [MHz]	Notes
Unpowered or in reset state	1149.1 or 1149.7	—	44.8	11.2	Assuming 64 MHz BD IRC with 30% tolerance
Powered, out of reset, PLL not engaged, system clock at IRC frequency	1149.1 or 1149.7	15.52	15.52	3.88	Assuming 16 MHz PD IRC with 3% tolerance
Powered, out of reset, PLL engaged	1149.1 or 1149.7	200	200	27.7	JTAG speed is limited by TCK negedge to TDO delay (14 ns) plus external setup to TCK posedge (4 ns)
Powered, out of reset, PLL engaged	LFAST/JTAGM	200	200	100	—

89.5 Reset logic

There are five sources of reset that are combined to generate the reset of system portions as described by [Table 2024](#):

1. BD power-on reset—asserts immediately upon assertion of BD POR, and remains asserted for 10 clock cycles after negation of POR.
2. PD reset—asserts two clock rising edges after assertion of reset signal coming from PD and remains asserted while the PD reset signal is asserted plus two clock rising edges after the PD reset signal is de-asserted (this reset source is reflected in the PD_RST bit of the BD_SIU_RSR register).
3. DCI FBD reset—asserts immediately when the FBD_RESET bit of the DCI Control Register is set, and remains asserted for one clock cycle after FBD_RESET is negated.
4. JTAG controller reset—asserts immediately when a reset request signal is received from the JTAG Controller by the execution of EXTEST, CLAMP, HIZ, TEST_LEAKAGE, EZPORT boundary instructions, and remains asserted for one clock cycle after the reset request is negated.
5. Nexus reset—asserts by DCI logic based on JCOMP state and JTAG source selected. DCI FBD also generates a Nexus reset.

In [Table 2024](#), an 'X' indicates that the reset source generates a reset signal to the specified block. JTAG mode only indicates that those blocks are reset by the corresponding reset sources only in the case where the debug interface is running through the JTAG external pins. When the debug interface is running through the LFAST interface or by software through JTAGM, those blocks are not reset by the mentioned reset sources.

Table 2024. Reset sources and functional blocks

Blocks	BD POR	PD reset	DCI FBD reset	JTAGC reset	Nexus reset
DCI	X	—	—	—	—
Z0 Nexus RWA	X	—	X	—	X

Table 2024. Reset sources and functional blocks

Blocks	BD POR	PD reset	DCI FBD reset	JTAGC reset	Nexus reset
BD XBAR	X	—	X	X	—
Overlay/Trace RAM	X	—	X	X	—
JTAGM	X	—	JTAG mode only	JTAG mode only	—
LFAST	X	—	JTAG mode only	JTAG mode only	—
NAR/DWPU	X	—	X	X	X
NAL	X	—	X	X	X
NAP	X	—	X	X	X
BD SIU	X	—	X	X	—
BD SIU GPIO registers	X	X	—	X	—
BD INIT flag	X	—	X	X	—
PD/BD interface logic	X	—	X	X	—

Note: The PD reset is not affected by the BD reset. The BD does not interfere with startup or shut down timing of the PD.

89.6 BD memory map

The BD memory map is contained within a 16 MB range. This entire range is accessed through the PD Flash BIU. The BD memory map is divided into two regions: overlay RAM and device registers.

The device register region allows the CPUs on the PD to access configuration registers of the peripherals in the BD, such as LFAST, JTAGM, XBAR configuration, etc. The device register region are divided into 16 KB regions for assignment to each peripheral block.

The memory map for the BD is shown below as an absolute address relative to the PD address map. Within the BD, only a 24-bit address bus is required. The upper 8 bits of the complete 32-bit address are decoded and removed by the PD PFLASH controller.

Table 2025. BD memory map

Start address	End address	Allocated size (KB)	Description
BD overlay RAM			
0x0C00_0000	0x0C0F_FFFF	1024	Extended overlay RAM
0x0C10_0000	0x0C1F_FFFF	1024	Reserved BD overlay RAM
0x0C20_0000	0x0C3F_FFFF	2048	Reserved BD overlay RAM
0x0C40_0000	0x0C5F_FFFF	2048	Reserved BD overlay RAM
0x0C60_0000	0x0C7F_FFFF	2048	Reserved BD overlay RAM
BD registers			

Table 2025. BD memory map (continued)

Start address	End address	Allocated size (KB)	Description
0x0C80_0000	0x0C80_3FFF	16	BD_SIUL2
0x0C80_4000	0x0C80_7FFF	16	LFAST
0x0C80_8000	0x0C80_BFFF	16	JTAGM
0x0C80_C000	0x0C80_FFFF	16	DWPU
0x0C81_0000	0x0C81_3FFF	16	Reserved
0x0C81_4000	0x0C81_7FFF	16	Reserved
0x0C81_8000	0x0C81_BFFF	16	Reserved
0x0C81_C000	0x0C81_FFFF	16	Reserved
0x0C82_0000	0x0CFF_FFFF	8064	Reserved

90 Buddy Device—Crossbar and RAM (BD_XBS_RAM)

Within the buddy device (BD), there is a crossbar switch that provides the required connections between multiple bus masters and the overlay RAM. For a diagram of the Emulation device (ED) debug architecture that shows the BD blocks and crossbar, see the Emulation device introduction chapter. This section provides details on the structure of this function.

Note: Throughout this description, the term “overlay RAM” is applied to the BD memory, even though this memory can be used for both overlay and debug trace information.

The 64-bit BD crossbar and RAM structure consists of:

- Input registers to capture transfers (address, attributes and write data) targeted at the overlay RAM or memory-mapped peripheral space
- A 3 x 5 crossbar switch to route requests from masters to either the overlay RAM or the system bus
 - Bus masters include the BD’s Nexus Aurora Router (BD-NAR), the production device’s flash controller (PFLASH) and the BD’s Nexus Read/Write Access Client (RWA)
 - Bus slaves include 4 equal-sized regions of the overlay RAM and a peripheral bridge (AIPS-Lite) bus controller
 - 160-bit RAM arrays (128 bits of data, 32 bits of end-to-end ECC)
 - End-to-End ECC32 address and data protection
 - ECC scheme based on 29-bit address and 32-bit data
 - Overlay RAM supports a 64-bit “early write” datapath and a 128-bit read datapath
- Output alignment logic and registers to capture read data being driven to the requesting master

The result is an efficient structure supporting concurrent references of the overlay RAM for calibration accesses from the PFLASH controller on the Production Device (PD) and debug references from the other masters. The crossbar and RAM support 200 MHz operation.

Note: Calibration reads from the PFLASH have the same timing characteristics as normal data reads from the flash array.

90.1 3 x 5 Crossbar switch and attached controllers

In the same manner as the comparable crossbar switches in the PD, the BD crossbar concurrently supports up to 3 simultaneous connections between the master and the slave ports. It implements a 64-bit “early write” datapath and a 128-bit read datapath and fully supports concurrent accesses between calibration reads and debug trace writes to different overlay RAM regions.

This crossbar implements fixed priority arbitration on a per slave port basis where the master number defines the relative priority in a reverse numeric order, that is, $m_0 > m_1 > m_2$ (PFLASH > BD-NAR > Nexus RWA). Each slave port is “parked” on the last master making a request.

The design supports {8, 16, 32, 64}-bit SINGLE reads, {8, 16, 32, 64}-bit SINGLE writes along with 64-bit WRAP4 burst reads and writes. Two consecutive sequential 128-bit accesses of the RAM are used (only) on the WRAP4 burst read.

90.2 BD_XBS_RAM End-to-End ECC32 Checkbit/Syndrome Coding Scheme

The e2eECC scheme used in the BD overlay RAM is a simplified derivative of the algorithm used in the Production Device. The only difference is the code is based on a 32-bit data field, but the same H-matrix is used with the appropriate 32-bit data upper or lower word zeroed.

Consider the following Boolean equation definitions:

$$e2eECC64[7:0] = ecc_addr[7:0] \wedge ecc_data_upper[7:0] \wedge ecc_data_lower[7:0]$$

where ecc_addr is H-matrix evaluation of the 29-bit access address ($addr[31:3]$), ecc_data_upper is the H-matrix evaluation of the upper data word ($data[63:32]$) and ecc_data_lower is the comparable evaluation of the lower data word ($data[31:0]$).

For the BD RAM definition, the following (re)definitions apply:

$$e2eECC32_upper[7:0] = ecc_addr[7:0] \wedge ecc_data_upper[7:0]$$

$$e2eECC32_lower[7:0] = ecc_addr[7:0] \wedge ecc_data_lower[7:0]$$

As a result, the PD's H-matrix is used for the BD RAM's e2eECC32 scheme with the appropriate 32 bit data word zeroed out:

$$e2eECC32_upper[7:0] = ecc_addr[7:0] \wedge ecc_data_upper[7:0]; data_lower[31:00] = 0$$

$$e2eECC32_lower[7:0] = ecc_addr[7:0] \wedge ecc_data_lower[7:0]; data_upper[63:32] = 0$$

In any case, the resulting ECC32 is calculated based on the appropriate 32 data bits plus 29 address bits (the upper bits of the 32-bit address field).

The complete H-matrix for this (101, 93) code is shown in [Table 2026](#). A '*' in [Table 2026](#) indicates the corresponding data or address bit is XOR'd to form the final checkbit value on the left. For 32-bit data writes, the table sections corresponding to D[63:32] or D[31:0], together with A[31:3] are logically summed (output of each table section is XOR'd) together to the final value driven on the $hwchckbit[7:0]$ outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 2026. e2eECC Basic H-Matrix Definition

Checkbits [7:0]	Data Bit ⁽¹⁾																																
	Byte 7								Byte 6								Byte 5								Byte 4								
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
7	*				*					*			*		*	*		*	*	*	*					*	*	*			*	*	
6			*	*			*	*				*		*			*				*				*			*	*			*	
5	*	*	*	*					*				*	*		*		*	*				*	*	*				*				
4	*		*			*		*				*		*			*			*			*	*			*				*		
3									*	*	*	*	*		*				*		*				*	*	*	*					

Table 2026. e2eECC Basic H-Matrix Definition (continued)

Checkbits [7:0]	Data Bit ⁽¹⁾																															
	Byte 7								Byte 6								Byte 5								Byte 4							
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
2					*		*				*		*				*	*				*	*							*		*
1		*		*		*		*	*	*	*							*		*		*		*					*			
0		*			*	*	*				*		*	*	*		*				*				*		*	*	*		*	
Checkbits [7:0]	Byte 3								Byte 2								Byte 1								Byte 0							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7					*		*				*		*				*	*			*	*			*				*	*	*	
6	*	*		*	*				*				*				*	*		*	*					*	*			*	*	
5			*	*			*	*	*			*	*	*					*		*	*	*	*								
4	*		*	*				*		*	*				*	*	*		*				*	*				*		*		
3	*	*	*			*					*	*		*	*	*			*			*	*	*	*	*	*	*	*	*	*	
2		*		*	*	*	*	*	*	*	*			*				*		*		*			*	*	*	*	*	*	*	
1		*						*		*		*	*	*	*			*			*				*	*		*	*	*	*	
0		*		*	*	*	*	*			*				*	*			*				*	*			*	*	*	*	*	
Checkbits [7:0]	Address Bit ⁽¹⁾																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3			
7		*		*		*		*	*	*		*	*	*	*	*	*	*		*	*	*	*	*	*	*	*	*	*			
6		*		*	*	*	*		*	*	*			*	*	*	*		*		*	*	*	*	*	*	*	*	*			
5		*	*	*			*	*			*	*		*		*	*	*	*		*	*	*	*	*	*	*	*	*			
4	*	*	*		*	*		*		*	*	*	*		*			*	*	*	*	*	*	*	*	*	*	*	*			
3	*		*		*	*	*		*		*	*	*			*		*	*	*	*	*	*		*	*	*	*	*			
2	*	*			*		*	*	*	*				*	*	*	*	*	*	*	*	*	*		*		*	*	*			
1	*		*	*		*	*		*	*	*	*	*	*	*		*	*	*	*	*	*	*				*	*	*			
0	*		*	*	*			*	*		*	*	*	*	*		*	*	*	*	*	*	*	*		*	*	*	*			

1. Bit numbering is AHB convention, bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

Figure 2162 through Figure 2165 show an alternative representation of the ECC32 encode process, written as a C language function.

Figure 2162. C language encodeECC32_upper function description (Part 1 of 4)

```

encodeEcc32_upper (addr, data_a2_is_zero)
    unsigned int      addr;                /* 32-bit byte address */
    unsigned int      data_a2_is_zero;     /* 32-bit data upper, a[2]=0 */
{
    unsigned int      addr_ecc;            /* 8 bits of ecc for address */
    unsigned int      ecc;                 /* 8 bits of ecc codeword */

    /* the following equation calculates the 8-bit wide ecc codeword by examining
    each addr or data bits and xor'ing the appropriate H-matrix value if the bit = 1 */

    addr_ecc
    = (((addr      >> 31) & 1) ? 0x1f : 0x0)      /* addr[31] */
    ^ (((addr      >> 30) & 1) ? 0xf4 : 0x0)      /* addr[30] */
    ^ (((addr      >> 29) & 1) ? 0x3b : 0x0)      /* addr[29] */
    ^ (((addr      >> 28) & 1) ? 0xe3 : 0x0)      /* addr[28] */
    ^ (((addr      >> 27) & 1) ? 0x5d : 0x0)      /* addr[27] */
    ^ (((addr      >> 26) & 1) ? 0xda : 0x0)      /* addr[26] */
    ^ (((addr      >> 25) & 1) ? 0x6e : 0x0)      /* addr[25] */
    ^ (((addr      >> 24) & 1) ? 0xb5 : 0x0)      /* addr[24] */

    ^ (((addr      >> 23) & 1) ? 0x8f : 0x0)      /* addr[23] */
    ^ (((addr      >> 22) & 1) ? 0xd6 : 0x0)      /* addr[22] */
    ^ (((addr      >> 21) & 1) ? 0x79 : 0x0)      /* addr[21] */
    ^ (((addr      >> 20) & 1) ? 0xba : 0x0)      /* addr[20] */
    ^ (((addr      >> 19) & 1) ? 0x9b : 0x0)      /* addr[19] */
    ^ (((addr      >> 18) & 1) ? 0xe5 : 0x0)      /* addr[18] */
    ^ (((addr      >> 17) & 1) ? 0x57 : 0x0)      /* addr[17] */
    ^ (((addr      >> 16) & 1) ? 0xec : 0x0)      /* addr[16] */

    ^ (((addr      >> 15) & 1) ? 0xc7 : 0x0)      /* addr[15] */
    ^ (((addr      >> 14) & 1) ? 0xae : 0x0)      /* addr[14] */
    ^ (((addr      >> 13) & 1) ? 0x67 : 0x0)      /* addr[13] */
    ^ (((addr      >> 12) & 1) ? 0x9d : 0x0)      /* addr[12] */
    ^ (((addr      >> 11) & 1) ? 0x5b : 0x0)      /* addr[11] */
    ^ (((addr      >> 10) & 1) ? 0xe6 : 0x0)      /* addr[10] */
    ^ (((addr      >>  9) & 1) ? 0x3e : 0x0)      /* addr[ 9] */
    ^ (((addr      >>  8) & 1) ? 0xf1 : 0x0)      /* addr[ 8] */

    ^ (((addr      >>  7) & 1) ? 0xdc : 0x0)      /* addr[ 7] */
    ^ (((addr      >>  6) & 1) ? 0xe9 : 0x0)      /* addr[ 6] */
    ^ (((addr      >>  5) & 1) ? 0x3d : 0x0)      /* addr[ 5] */
    ^ (((addr      >>  4) & 1) ? 0xf2 : 0x0)      /* addr[ 4] */
    ^ (((addr      >>  3) & 1) ? 0x2f : 0x0);     /* addr[ 3] */

```

Figure 2163. C language encodeECC32_upper function description (Part 2 of 4)

```

ecc = (((data_a2_is_zero >> 31) & 1) ? 0xb0 : 0x0)      /* data[63] */
^ (((data_a2_is_zero >> 30) & 1) ? 0x23 : 0x0)        /* data[62] */
^ (((data_a2_is_zero >> 29) & 1) ? 0x70 : 0x0)        /* data[61] */
^ (((data_a2_is_zero >> 28) & 1) ? 0x62 : 0x0)        /* data[60] */
^ (((data_a2_is_zero >> 27) & 1) ? 0x85 : 0x0)        /* data[59] */
^ (((data_a2_is_zero >> 26) & 1) ? 0x13 : 0x0)        /* data[58] */
^ (((data_a2_is_zero >> 25) & 1) ? 0x45 : 0x0)        /* data[57] */
^ (((data_a2_is_zero >> 24) & 1) ? 0x52 : 0x0)        /* data[56] */

^ (((data_a2_is_zero >> 23) & 1) ? 0x2a : 0x0)        /* data[55] */
^ (((data_a2_is_zero >> 22) & 1) ? 0x8a : 0x0)        /* data[54] */
^ (((data_a2_is_zero >> 21) & 1) ? 0x0b : 0x0)        /* data[53] */
^ (((data_a2_is_zero >> 20) & 1) ? 0x0e : 0x0)        /* data[52] */
^ (((data_a2_is_zero >> 19) & 1) ? 0xf8 : 0x0)        /* data[51] */
^ (((data_a2_is_zero >> 18) & 1) ? 0x25 : 0x0)        /* data[50] */
^ (((data_a2_is_zero >> 17) & 1) ? 0xd9 : 0x0)        /* data[49] */
^ (((data_a2_is_zero >> 16) & 1) ? 0xa1 : 0x0)        /* data[48] */

^ (((data_a2_is_zero >> 15) & 1) ? 0x54 : 0x0)        /* data[47] */
^ (((data_a2_is_zero >> 14) & 1) ? 0xa7 : 0x0)        /* data[46] */
^ (((data_a2_is_zero >> 13) & 1) ? 0xa8 : 0x0)        /* data[45] */
^ (((data_a2_is_zero >> 12) & 1) ? 0x92 : 0x0)        /* data[44] */
^ (((data_a2_is_zero >> 11) & 1) ? 0xc8 : 0x0)        /* data[43] */
^ (((data_a2_is_zero >> 10) & 1) ? 0x07 : 0x0)        /* data[42] */
^ (((data_a2_is_zero >> 9) & 1) ? 0x34 : 0x0)         /* data[41] */
^ (((data_a2_is_zero >> 8) & 1) ? 0x32 : 0x0)         /* data[40] */

^ (((data_a2_is_zero >> 7) & 1) ? 0x68 : 0x0)         /* data[39] */
^ (((data_a2_is_zero >> 6) & 1) ? 0x89 : 0x0)         /* data[38] */
^ (((data_a2_is_zero >> 5) & 1) ? 0x98 : 0x0)         /* data[37] */
^ (((data_a2_is_zero >> 4) & 1) ? 0x49 : 0x0)         /* data[36] */
^ (((data_a2_is_zero >> 3) & 1) ? 0x61 : 0x0)         /* data[35] */
^ (((data_a2_is_zero >> 2) & 1) ? 0x86 : 0x0)         /* data[34] */
^ (((data_a2_is_zero >> 1) & 1) ? 0x91 : 0x0)         /* data[33] */
^ ((data_a2_is_zero & 1) ? 0x46 : 0x0);              /* data[32] */

return(ecc);
}

```

Figure 2164. C language encodeECC32_upper function description (Part 3 of 4)

```

C Language encodeECC32_lower Function Description
encodeEcc32_lower (addr, data_a2_is_one)
    unsigned int      addr;                /* 32-bit byte address */
    unsigned int      data_a2_is_one;      /* 32-bit data lower, a[2]=1 */
{
    unsigned int      addr_ecc;            /* 8 bits of ecc for address */
    unsigned int      ecc;                 /* 8 bits of ecc codeword */
    /* the following equation calculates the 8-bit wide ecc codeword by examining
    each addr or data bits and xor'ing the appropriate H-matrix value if the bit = 1 */
    addr_ecc
    = ((addr          >> 31) & 1) ? 0x1f : 0x0)      /* addr[31] */
    ^ ((addr          >> 30) & 1) ? 0xf4 : 0x0)      /* addr[30] */
    ^ ((addr          >> 29) & 1) ? 0x3b : 0x0)      /* addr[29] */
    ^ ((addr          >> 28) & 1) ? 0xe3 : 0x0)      /* addr[28] */
    ^ ((addr          >> 27) & 1) ? 0x5d : 0x0)      /* addr[27] */
    ^ ((addr          >> 26) & 1) ? 0xda : 0x0)      /* addr[26] */
    ^ ((addr          >> 25) & 1) ? 0x6e : 0x0)      /* addr[25] */
    ^ ((addr          >> 24) & 1) ? 0xb5 : 0x0)      /* addr[24] */

    ^ ((addr          >> 23) & 1) ? 0x8f : 0x0)      /* addr[23] */
    ^ ((addr          >> 22) & 1) ? 0xd6 : 0x0)      /* addr[22] */
    ^ ((addr          >> 21) & 1) ? 0x79 : 0x0)      /* addr[21] */
    ^ ((addr          >> 20) & 1) ? 0xba : 0x0)      /* addr[20] */
    ^ ((addr          >> 19) & 1) ? 0x9b : 0x0)      /* addr[19] */
    ^ ((addr          >> 18) & 1) ? 0xe5 : 0x0)      /* addr[18] */
    ^ ((addr          >> 17) & 1) ? 0x57 : 0x0)      /* addr[17] */
    ^ ((addr          >> 16) & 1) ? 0xec : 0x0)      /* addr[16] */

    ^ ((addr          >> 15) & 1) ? 0xc7 : 0x0)      /* addr[15] */
    ^ ((addr          >> 14) & 1) ? 0xae : 0x0)      /* addr[14] */
    ^ ((addr          >> 13) & 1) ? 0x67 : 0x0)      /* addr[13] */
    ^ ((addr          >> 12) & 1) ? 0x9d : 0x0)      /* addr[12] */
    ^ ((addr          >> 11) & 1) ? 0x5b : 0x0)      /* addr[11] */
    ^ ((addr          >> 10) & 1) ? 0xe6 : 0x0)      /* addr[10] */
    ^ ((addr          >>  9) & 1) ? 0x3e : 0x0)      /* addr[ 9] */
    ^ ((addr          >>  8) & 1) ? 0xf1 : 0x0)      /* addr[ 8] */

    ^ ((addr          >>  7) & 1) ? 0xdc : 0x0)      /* addr[ 7] */
    ^ ((addr          >>  6) & 1) ? 0xe9 : 0x0)      /* addr[ 6] */
    ^ ((addr          >>  5) & 1) ? 0x3d : 0x0)      /* addr[ 5] */
    ^ ((addr          >>  4) & 1) ? 0xf2 : 0x0)      /* addr[ 4] */
    ^ ((addr          >>  3) & 1) ? 0x2f : 0x0);      /* addr[ 3] */

```

Figure 2165. C language encodeECC32_upper function description (Part 4 of 4)

```

ecc = (((data_a2_is_one >> 31) & 1) ? 0x58 : 0x0)      /* data[31] */
^ (((data_a2_is_one >> 30) & 1) ? 0x4f : 0x0)      /* data[30] */
^ (((data_a2_is_one >> 29) & 1) ? 0x38 : 0x0)      /* data[29] */
^ (((data_a2_is_one >> 28) & 1) ? 0x75 : 0x0)      /* data[28] */
^ (((data_a2_is_one >> 27) & 1) ? 0xc4 : 0x0)      /* data[27] */
^ (((data_a2_is_one >> 26) & 1) ? 0x0d : 0x0)      /* data[26] */
^ (((data_a2_is_one >> 25) & 1) ? 0xa4 : 0x0)      /* data[25] */
^ (((data_a2_is_one >> 24) & 1) ? 0x37 : 0x0)      /* data[24] */

^ (((data_a2_is_one >> 23) & 1) ? 0x64 : 0x0)      /* data[23] */
^ (((data_a2_is_one >> 22) & 1) ? 0x16 : 0x0)      /* data[22] */
^ (((data_a2_is_one >> 21) & 1) ? 0x94 : 0x0)      /* data[21] */
^ (((data_a2_is_one >> 20) & 1) ? 0x29 : 0x0)      /* data[20] */
^ (((data_a2_is_one >> 19) & 1) ? 0xea : 0x0)      /* data[19] */
^ (((data_a2_is_one >> 18) & 1) ? 0x26 : 0x0)      /* data[18] */
^ (((data_a2_is_one >> 17) & 1) ? 0x1a : 0x0)      /* data[17] */
^ (((data_a2_is_one >> 16) & 1) ? 0x19 : 0x0)      /* data[16] */

^ (((data_a2_is_one >> 15) & 1) ? 0xd0 : 0x0)      /* data[15] */
^ (((data_a2_is_one >> 14) & 1) ? 0xc2 : 0x0)      /* data[14] */
^ (((data_a2_is_one >> 13) & 1) ? 0x2c : 0x0)      /* data[13] */
^ (((data_a2_is_one >> 12) & 1) ? 0x51 : 0x0)      /* data[12] */
^ (((data_a2_is_one >> 11) & 1) ? 0xe0 : 0x0)      /* data[11] */
^ (((data_a2_is_one >> 10) & 1) ? 0xa2 : 0x0)      /* data[10] */
^ (((data_a2_is_one >> 9) & 1) ? 0x1c : 0x0)      /* data[ 9] */
^ (((data_a2_is_one >> 8) & 1) ? 0x31 : 0x0)      /* data[ 8] */

^ (((data_a2_is_one >> 7) & 1) ? 0x8c : 0x0)      /* data[ 7] */
^ (((data_a2_is_one >> 6) & 1) ? 0x4a : 0x0)      /* data[ 6] */
^ (((data_a2_is_one >> 5) & 1) ? 0x4c : 0x0)      /* data[ 5] */
^ (((data_a2_is_one >> 4) & 1) ? 0x15 : 0x0)      /* data[ 4] */
^ (((data_a2_is_one >> 3) & 1) ? 0x83 : 0x0)      /* data[ 3] */
^ (((data_a2_is_one >> 2) & 1) ? 0x9e : 0x0)      /* data[ 2] */
^ (((data_a2_is_one >> 1) & 1) ? 0x43 : 0x0)      /* data[ 1] */
^ ((data_a2_is_one & 1) ? 0xc1 : 0x0);          /* data[ 0] */

return(ecc);
}

```

As the ECC syndrome is calculated on a read operation by applying the H-matrix to the data plus the checkbits, an all zero syndrome indicates an error free operation. If the generated syndrome value is non-zero and matches one of the H-matrix values associated with the data or checkbits, it represents a single-bit error correction case and the specific bit is

complemented to produce the correct data value. If the syndrome value matches one of the H-matrix values associated with the address bits, the opposite data word, or is an even weight value, or represents an unused odd weight value, a non-correctable ECC event has been detected and the appropriate error termination response is initiated.

91 Buddy Device—System Integration Unit Lite (BD_SIUL2)

91.1 Introduction

The Buddy Device—System Integration Unit Lite2 (BD_SIUL2) provides control over the electrical pads of the buddy device (BD), as well as implementing various SoC functions that are not included in other modules.

The main functions supported by BD_SIUL2 are:

- Device identification
- Device status reporting
- Pad control and signal muxing

It should be noted that some other system level features are located in the BD DCI block including:

- Device identification available via the JTAG accessible Nexus Device ID (DID) register
- Reporting of reset status via the JTAG accessible reset status register
- Control of production device (PD) and BD reset via JTAG accessible reset control register

These registers are also indirectly accessible by software via BD_JTAGM in software debug mode.

91.2 Device identification

The BD_SIUL2 includes a device ID register (DIDR) that identifies the BD type and version, allowing a connected tool or development/calibration software running on a connected PD to determine what debug or calibration resources are available. The DIDR uses similar formatting to the MCU ID registers (MIDR1, MIDR2) included in the SIUL2 module.

This information is made available via the BD SIU rather than via the PD because it is either not available via the PD (for example, information on BD-specific resources, such as extended overlay RAM size) or because it is useful for the tool to be able to access the information before the PD is powered up.

Information available includes:

- An ID code that can be used to uniquely identify the BD type
- A version field that is changed for every major mask revision of the BD
- A parameterized field that indicates the amount of extended overlay RAM included
 - The field coding allows indication of up to 4 MB of RAM, with resolution of 256 KB steps

It should be noted that in addition to information available in the DIDR, a connected tool can also retrieve identification information from other sources including:

- The JTAG ID returned by the buddy DCI/JTAGC module, which is different than the JTAG ID returned by a standard PD DCI/JTAGC
- The MIDR register included in the PD SIUL2 module, which includes information on the PD device type, version and package type (including identification of emulation and debug device package variants)

91.3 Device status information

91.3.1 Initialization status

The BD SIU includes a flag (BD_INIT) that can be used by the connected PD or any connected debug or calibration tools to track the status of the BD.

The main expected usage mode for this bit is to allow the Boot Assist Flash (BAF) or user calibration software on the PD to determine whether the BD has lost power since it was last initialized. A typical usage sequence for BD_INIT is shown below.

- At BD power on reset (POR), BD_INIT is cleared automatically
 - BD_INIT is not cleared by external BD reset input
- A calibration tool or calibration software initializes the BD (for example, writing BD overlay RAM with calibration data and remap configuration)
- The calibration tool or calibration software sets BD_INIT
- After PD POR, the BAF executes and reads BD_INIT
 - If BD_INIT is clear, the BD overlay is assumed to be uninitialized and is not used for calibration remap
 - If BD_INIT is set, the BD overlay is assumed to be initialized and can be used for calibration remap. (Other tests to verify data integrity may be used.)

BD_INIT is cleared by BD POR, and can be set or cleared via the debug interface or writes from the PD. BD_INIT is only used to detect full power down events resulting in a subsequent POR. It does not detect brown out drops in the supply voltage.

91.3.2 PD reset status

The IPS accessible BD_SIUL2_RSR register shows what stage of reset processing the connected PD is in, and whether the PD is powered down or not. This is functionally different from the software accessible registers in the PD reset generation module that indicate which reset source has caused the last PD reset.

91.4 Memory map

[Table 2027](#) gives an overview of the BD_SIUL2 registers.

Table 2027. BD_SIUL2 register summary

Offset	Register	Access	Reset Value	Section/Page
0x0000	BD_SIUL2 Device Identification Register 1 (BD_SIUL2_DIDR1)	R	0x57BD_0000	91.4.1.1/3740
0x0004	BD_SIUL2 Device Identification Register 2 (BD_SIUL2_DIDR2)	R	0x0003_0100	91.4.1.2/3741
0x0100	BD_SIUL2 Initialization Status Register (BD_SIUL2_ISR)	R/W	0x0000_0000	91.4.1.3/3742
0x0104	BD_SIUL2 Reset Status Register (BD_SIUL2_RSR)	R	0x0000_0000	91.4.1.4/3743

91.4.1 Register descriptions

This section describes in address order all the BD_SIUL2 registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Note: All registers support 32-bit accesses only.

91.4.1.1 BD_SIUL2 Device Identification Register 1 (BD_SIUL2_DIDR1)

The BD_SIUL2_DIDR1 contains the part number and mask number.

Figure 2166. BD_SIUL2 Device Identification Register 1 (BD_SIUL2_DIDR1)

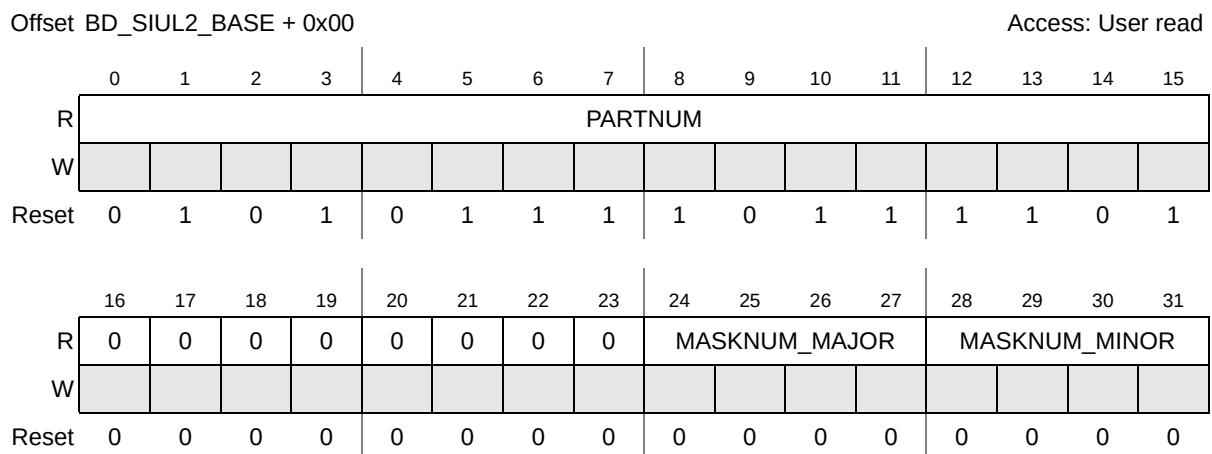


Table 2028. BD_SIUL2_DIDR1 Bit Field Descriptions

Field	Description
0–15 PARTNUM	Device Part Number. Read-only, mask programmed part number of the device. Reads 0x57BD.
16–23	Reserved.
24–27 MASKNUM_MAJOR	Major Mask Revision Number. Read-only, mask programmed mask number of the device. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.
28–31 MASKNUM_MINOR	Minor Mask Revision Number. Read-only, mask programmed mask number of the MCU. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.

91.4.1.2 BD_SIUL2 Device Identification Register 2 (BD_SIUL2_DIDR2)

The BD_SIUL2_DIDR2 contains a complement to the part number and amount of RAM available.

Figure 2167. BD_SIUL2 Device Identification Register 2 (BD_SIUL2_DIDR2)

Offset BD_SIUL2_BASE + 0x04												Access: User read				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	RAM_SIZE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2029. BD_SIUL2_DIDR2 Bit Field Descriptions

Field	Description
0–11	Reserved.
12–15 RAM_SIZE	RAM Size. Read-only, indicates the amount of overlay RAM available in the device, with resolution of 256K byte steps 0000 0.25M byte. 0001 0.5M byte. 0010 0.75M byte. 0011 1M byte. 0100 1.25M byte. 0101 1.5M byte. 0110 1.75M byte. 0111 2M byte. 1000 2.25M byte. 1001 2.5M byte. 1010 2.75M byte. 1011 3M byte. 1100 3.25M byte. 1101 3.5M byte. 1110 3.75M byte. 1111 4M byte.
16–23 PARTNUM	Device Part Number. Read-only, mask programmed complement to the part number of the device. For the BD 1M byte part with part number 57BD1, BD_SIUL2_DIDR1.PARTNUM=57BD, and BD_SIUL2_DIDR2.PARTNUM=1. 01 1 (BD1M) 02 2 (BD2M)
24–31	Reserved.

91.4.1.3 BD_SIUL2 Initialization Status Register (BD_SIUL2_ISR)

The BD_SIUL2_ISR contains a flag that can be used by the PD or any connected debug or calibration tool to track the status of the BD. This register is reset by POR.

The main expected use case for this flag is to allow software on the PD to determine whether the BD has lost power since it was last initialized. A typical sequence is:

- At power on reset (POR), the flag is cleared automatically (not affected by software reset)
- A calibration tool or PD calibration software initializes the BD (for example, writing BD overlay RAM with calibration data and remap configuration)
- The calibration tool or PD calibration software sets the BD_INIT flag
- After PD POR, the PD boot software executes and reads BD_INIT
 - If BD_INIT is clear, BD overlay is assumed to be uninitialized and is not used for calibration remap
 - If BD_INIT is set, BD overlay is assumed to be initialized and can be used for calibration remap (other tests to verify data integrity may be used)

BD_INIT is cleared by BD POR, and can be set or cleared via writes from the PD or from the Nexus RWA.

Figure 2168. BD_SIUL2 Initialization Status Register (BD_SIUL2_ISR)

Offset BD_SIUL2_BASE + 0x100												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BD_INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 2030. BD_SIUL2_ISR Bit Field Descriptions

Field	Description
0–30	Reserved.
31 BD_INIT	Initialization Flag. This is a read/write bit that can be used to indicate that the BD was initialized. Software reset does not affect the flag. Power-on reset clear the flag. 0 BD not initialized. 1 BD initialized.

91.4.1.4 BD_SIUL2 Reset Status Register (BD_SIUL2_RSR)

The BD_SIUL2_RSR contains reset status information about the PD.

Figure 2169. BD_SIUL2 Reset Status Register (BD_SIUL2_RSR)

Offset BD_SIUL2_BASE + 0x104												Access: User read				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	PD_RST	PD_TRIM	PD_POR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2031. BD_SIUL2_RSR Bit Field Descriptions

Field	Description
0–28	Reserved.
29 PD_RST	PD Reset Flag. Read-only, indicates if PD has finished the reset sequence. 0 PD reset sequence not completed. 1 PD reset sequence completed.
30 PD_TRIM	PD Trim Flag. Read-only, indicates if PD has finished trimming phase during the reset sequence. 0 PD trimming phase not completed. 1 PD trimming phase completed.
31 PD_POR	PD POR Flag. Read-only, indicates whether PD is powered down or not. 0 PD is powered down or power on reset has not completed. 1 PD is powered and power on reset has completed.

92 Buddy Device—Debug and Calibration Interface (DCI)

92.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

To provide high-speed calibration capability, the DCI also provides a mechanism to use 5-pin LFAST debug tool which shares its pins with the standard JTAG interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features in the Emulation and Debug Device (ED) that includes features in the Production device (PD) as well as features in the Buddy device (BD). There are two DCIs on the ED which are designated PD DCI and BD DCI.

92.1.1 Features

The ED DCI features are the following:

- Debug mode enable control for connected tool or software
- Dual DCI system for multi-device clients
- Port sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

92.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Buddy device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using LFAST or software debug mode.
 - When using LFAST with a connected calibration/debug tool, JTAG packets are received from the tool via LFAST and output JTAG packets are transmitted back to the tool. The LFAST's 5-pin interface consists of an LVDS pair for transmit signals, an LVDS pair of receive signals and a reference clock output signal.
 - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.
- I.JTAG— JTAG signals interconnecting the two DCI implementations in the ED coming from the intra-die pads.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode. The LFAST-based debug interface shares its five pins with the standard JTAG pins.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

As the LFAST-based debug interface shares its five pins with JTAG pins, the DCI provides a safe switching logic to switch to LFAST-based debug mode.

The ED DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as GTM/SPU, etc. The ED DCI also provides the Event Out (EVTO) signals to the debug tool in case of the occurrence of any internal debug event.

[Figure 2170](#) shows the DCI block diagram.

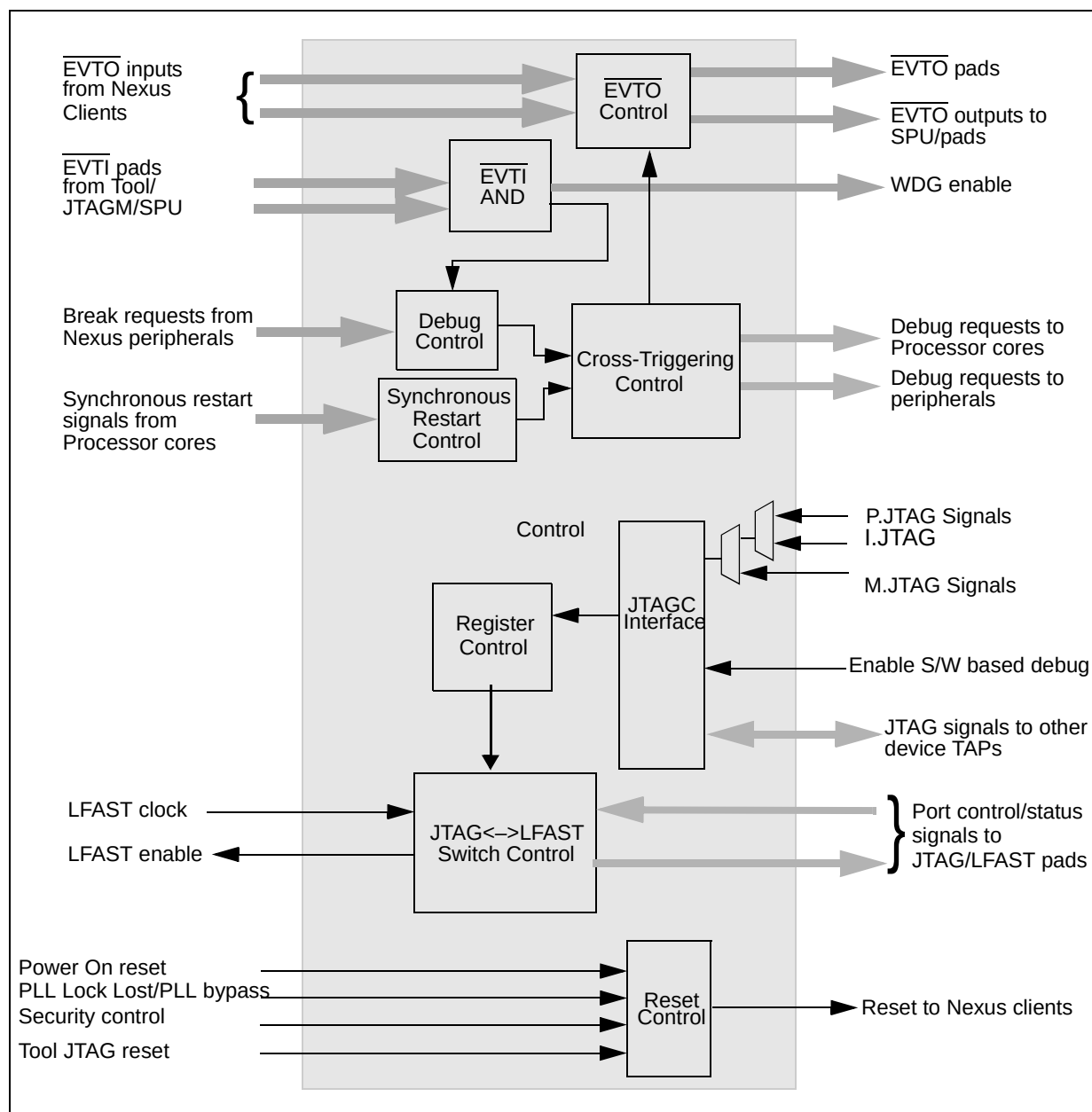


Figure 2170. DCI block diagram

When the DCI is used in an ED package (PD and the BD together) its configuration is shown as in [Figure 2171](#). In the ED package, a debug tool is connected to the BD pins and connects to the PD with the I.JTAG interface which are routed using inter-die pads.

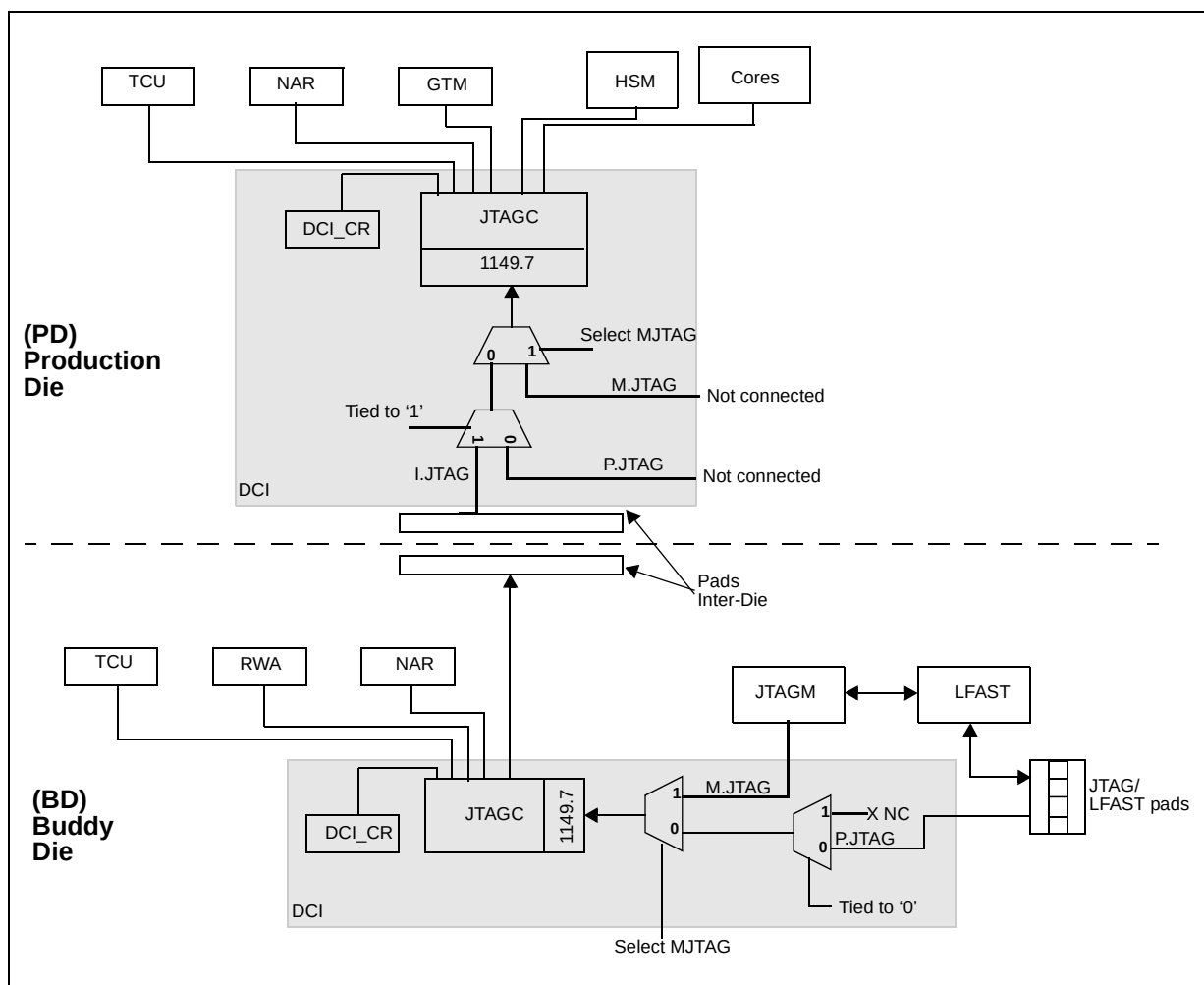


Figure 2171. DCI in an emulation configuration

92.1.3 Operating modes

Selection between various JTAG sources is done using a JTAG register bit, DCI_CR[EN_LFAST], and a JTAGM register bit, JTAGM_MCR[DTM]. The JTAG bit, DCI_CR[EN_LFAST], can only be programmed using JTAG instructions and is used to select standard JTAG or LFAST debug tool. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when either DCI_CR[EN_LFAST] or JTAGM_MCR[DTM] is set.

Table 2032 shows a summary of the configuration options.

Table 2032. DCI JTAG source selection

DCI_CR[EN_LFAST]	JTAGM_MCR[DTM]	DCI Source	JCOMP
0	0	P.JTAG (Standard JTAG)	Asserted
1	x	M.JTAG (LFAST)	Asserted
x	1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated. In the ED, the BD DCI is reset with BD POR/Low voltage detection on BD_VDD.

The M.JTAG provides an alternate interface to the debug logic.

When enabled, it allows high speed debug through the BD LFAST and BD JTAGM modules. The high speed link uses LFAST commands that are translated to JTAG commands in the JTAGM module. The selection between LFAST based debug or software based debug is managed in the JTAGM. However, it is important to note that the dynamic switching between tool-based debug and software debug is not allowed. Software based debug is done only in those cases when an external tool is not present.

An LFAST based tool starts in JTAG mode and then switches to LFAST mode. This switching is managed through a control set of operations which is described in [Section 92.1.3.1, Port sharing between JTAG and LFAST modes](#). During the switch from JTAG to LFAST and back, the DCI uses a latched value of JCOMP to prevent all debug configuration reset.

The LFAST uses LVDS ports for LFAST communication and the JTAG ports uses CMOS level GPIOs. These signals are double-bonded to share the same pins at the tool interface. [Table 2033](#) shows the port mapping.

When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

Table 2033. DCI LFAST/JTAG port sharing

JTAG pad	Type in JTAG mode	LFAST pad	Type in LFAST mode
TDI	Input	LVDS TxN	Output
TMS	Input	LVDS TxP	Output
TDO	Output	LVDS RxP	Input
JCOMP	Input	LVDS RxN	Input
TCK	Input	DRCLK	Output

[Figure 2172](#) shows the DCI mode switching operation.

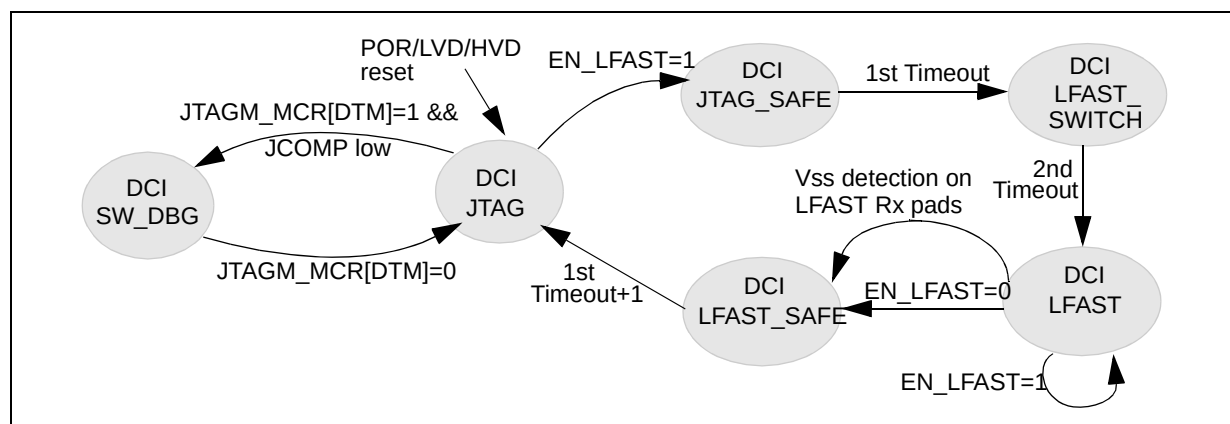


Figure 2172. DCI operating modes state diagram

Note: .JTAGM_MCR[jtagm_JCOMP] should not be asserted in LFAST mode to avoid any undesired behavior.

Table 2034 contains the operating mode descriptions.

Table 2034. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG or I.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.
DCI_JTAG_SAFE	SAFE_JTAG state. When an external tool sets the DCI_CR[EN_LFAST] bit, this indicates that the external tool is going to switch to the LFAST mode. The external tool puts the DCI in Run-Test-Idle state to start the switching operation. To enable safe switching operation, the DCI enters the JTAG_SAFE state where it performs the following operations: – JCOMP is latched inside to control until switching gets completed – The DCI releases control of the JTAG ports and uses high on TMS inside, thus keeping the DCI in the Run-Test-Idle state – An internal counter starts to count to a tool specified value as programmed in DCI_CR[SWITCH_CNTR1].
DCI_LFAST_SWITCH	LFAST port switching state. In this state, the DCI asserts the enable signal to indicate to the LFAST that it is now ready for LFAST port switching. In this mode, the LFAST Rx pads are enabled. The switch counter starts another count operation to count to the DCI_CR[SWITCH_CNTR2] value.

Table 2034. DCI operating modes (continued)

Mode	Description
DCI_LFAST	LFAST mode with escape mode enabled. This mode indicates that the switching is now completed and the debug tool is operating in the LFAST mode. However, there is another feature called Escape Mode, which enables the LFAST pads to detect any disconnection fault on its Rx pads. When these pads are working in Escape Mode, a sensing logic is enabled in the pad which detects and flags any out of common mode voltage range for the Rx pads. If this flag is asserted, it is used to force the DCI to go to LFAST_SAFE mode to ensure safe switching to the JTAG state.
DCI_LFAST_SAFE	LFAST_SAFE mode. While the tool is switching from LFAST mode to JTAG mode, it resets the DCI_CR[EN_LFAST] bit. The external tool puts the DCI in the Run-Test-Idle state to start the switching operation. To account for the switching time for the pads, the DCI enters in a safe LFAST state. It then restarts a counter to count to a tool specified value in DCI_CR[SWITCH_CNTR1], and disables both the LFAST function on the pads and the LFAST module.

Figure 2173 shows a block diagram of the JTAG/LFAST connections.

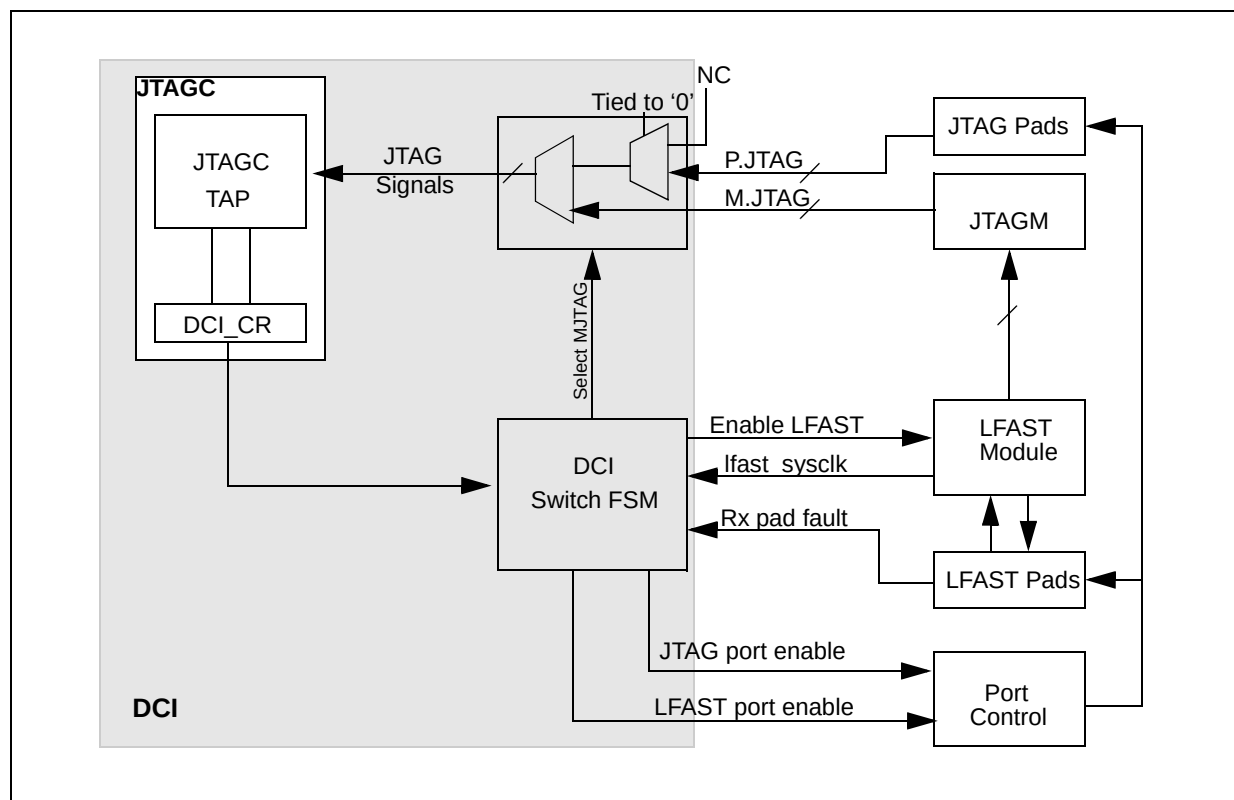


Figure 2173. DCI JTAG/LFAST switch integration

92.1.3.1 Port sharing between JTAG and LFAST modes

By default the DCI is configured in JTAG mode by using the standard JTAG port (P.JTAG). Therefore, the initial configuration is done using low speed JTAG signals. In order to switch

from JTAG mode to LFAST mode, the tool accesses the following fields in the DCI control register (DCI_CR):

- EN_LFAST—Bit to indicate enable/disable LFAST mode
- SWITCH_CNTR1 and SWITCH_CNTR2—Bit fields to program a count value that corresponds to the delay for switching the pin functions when changing modes.

As the JTAG to LFAST mode switching reflects change in port input/output behavior, it is done through a controlled procedure. The DCI_CR is only accessible through JTAG sequences. While switching from JTAG to LFAST mode, the tool programs this register via P.JTAG to enable switching to LFAST mode. As the pins are shared between JTAG and LFAST functions, a programmable delay based on switch counters is required which allows a safe switching for the MCU as well as for the tool to change its mode. In LFAST mode, the tool can write this register to switch back from high speed debug to low speed debug. Again, the similar switching delay mechanism is used. The switching schematic is shown in [Figure 2174](#).

A programmable switching delay and tri-stating port pins during switching allow a connected tool to safely tune the device turn-around of pin functions from JTAG mode to LFAST mode and vice versa. These delays are programmed according to the time required by the tool to change its operation from JTAG mode to LFAST and vice versa. This reduces the possibility of device damage during switching.

The programmable counter uses the LFAST system clock as the clock source.

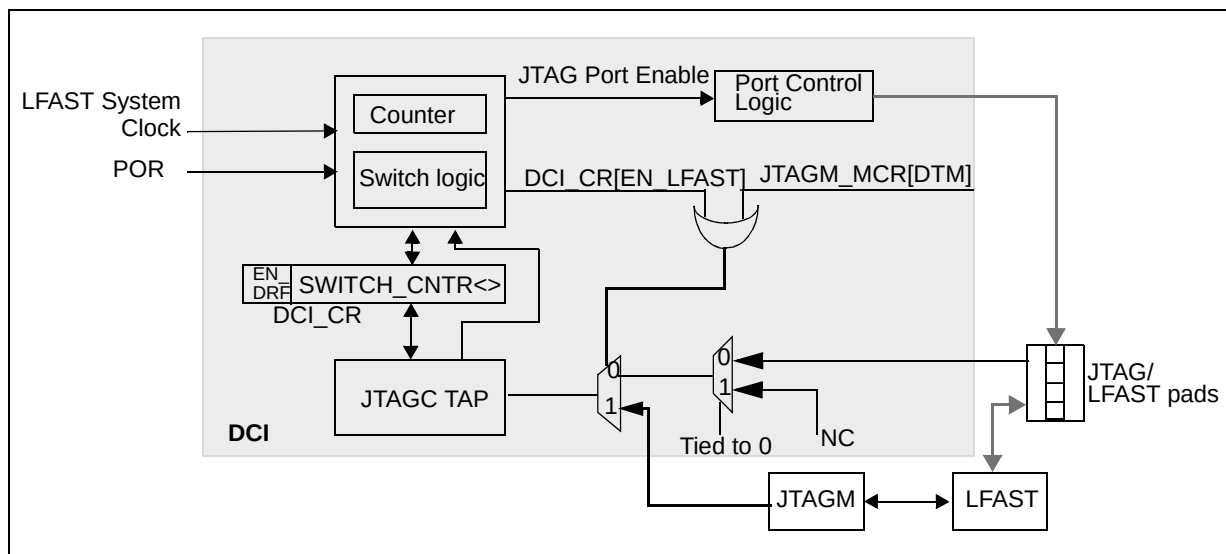


Figure 2174. DCI switching between JTAG and LFAST modes

92.1.3.1.1 Switching from JTAG mode to LFAST mode

The steps to switch operation from JTAG to LFAST mode are as follows:

1. System starts in port JTAG mode after power-on-reset.
2. External tool programs the following bits in the DCI_CR:
 - EN_LFAST
 - SWITCH_CNTR1
 - SWITCH_CNTR2
3. Tool puts JTAG TAP in the Run-Test-Idle mode.
4. An internal finite state machine (FSM) starts the switching operation using the LFAST reference clock. The switching process takes four LFAST reference clocks to start, and an external tool should not change the JTAG signals during this period. The DCI forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. The DCI also triggers a switching counter to allow for some time for the tool to change its mode from JTAG to LFAST.
5. After the counter counts to the DCI_CR[SWITCH_CNTR1] value, the LFAST IP and LFAST pads are enabled. Then the counter is reset. Also, internally JTAGM is selected as the JTAG source for the DCI.
6. Counter counts to the DCI_CR[SWITCH_CNTR2] value to enable the Escape mode feature for the LFAST Rx pads. This time allows the tool to settle down in LFAST mode.
7. Tool starts LFAST transmission when it detects pulses on the DRCLK pin. Tool now can use the LFAST to program the JTAGM to generate JTAG sequences.

[Figure 2175](#) shows the timing sequence.

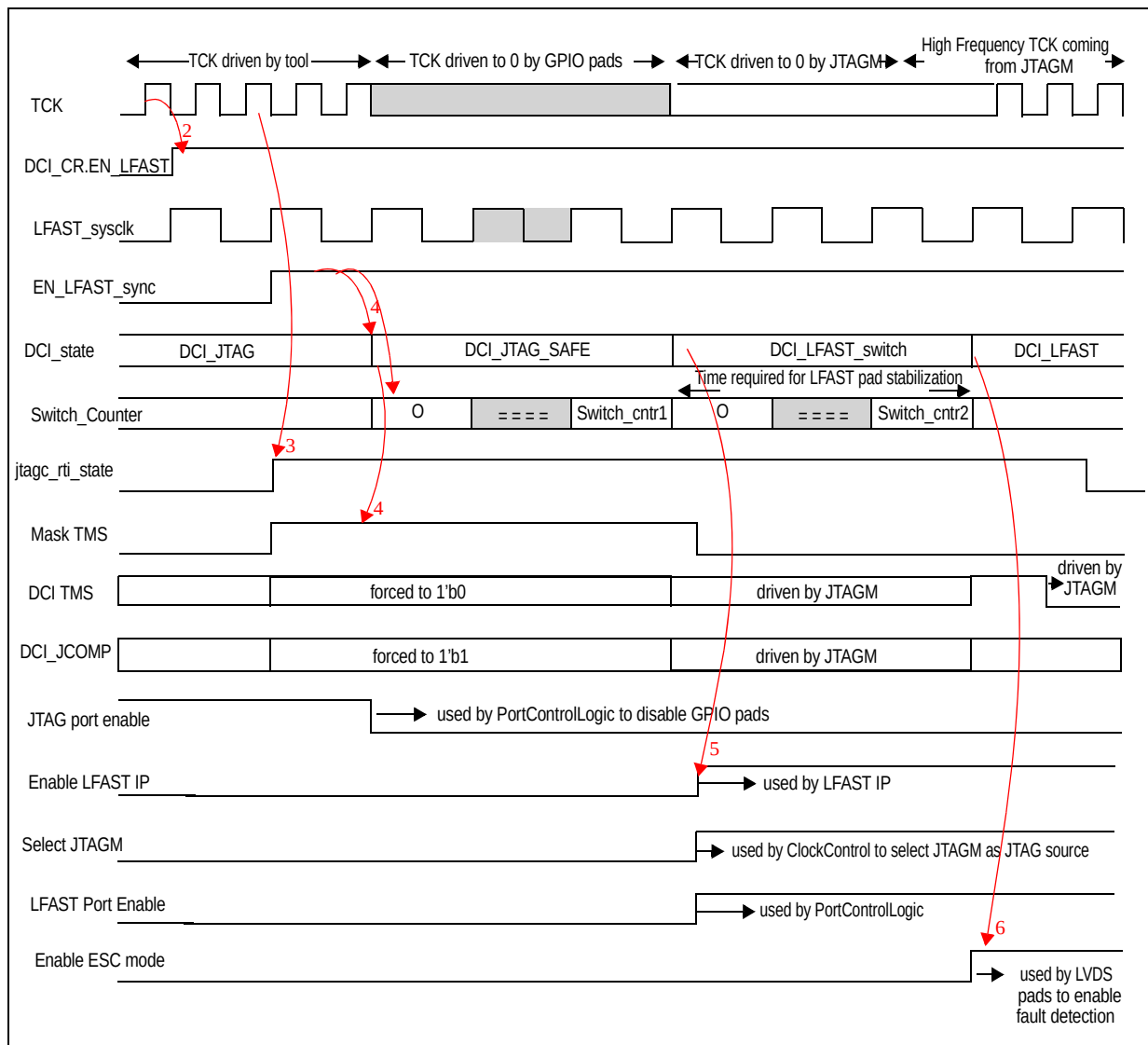


Figure 2175. DCI JTAG to LFAST switching diagram

92.1.3.1.2 Switching from LFAST mode to JTAG mode

The steps to switch operation from LFAST to JTAG mode are as follows:

1. External tool programs the DCI_CR to reset the EN_LFAST bit and SWITCH_CNTR1 for the switching delay.
2. Tool puts JTAG TAP in the Run-Test-Idle mode.
3. An internal FSM starts switching operation. It forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. It also triggers a switching counter to count to DCI_CR[SWITCH_CNTR1] to allow for the tool to change its mode from LFAST to JTAG.
4. After the counter overflows, the JTAG pads are enabled and JTAGM releases its control for the DCI JTAG source.

Table 2038 and Table 2040 summarize the MCU and tool behaviors. Figure 2176 shows the timing sequence.

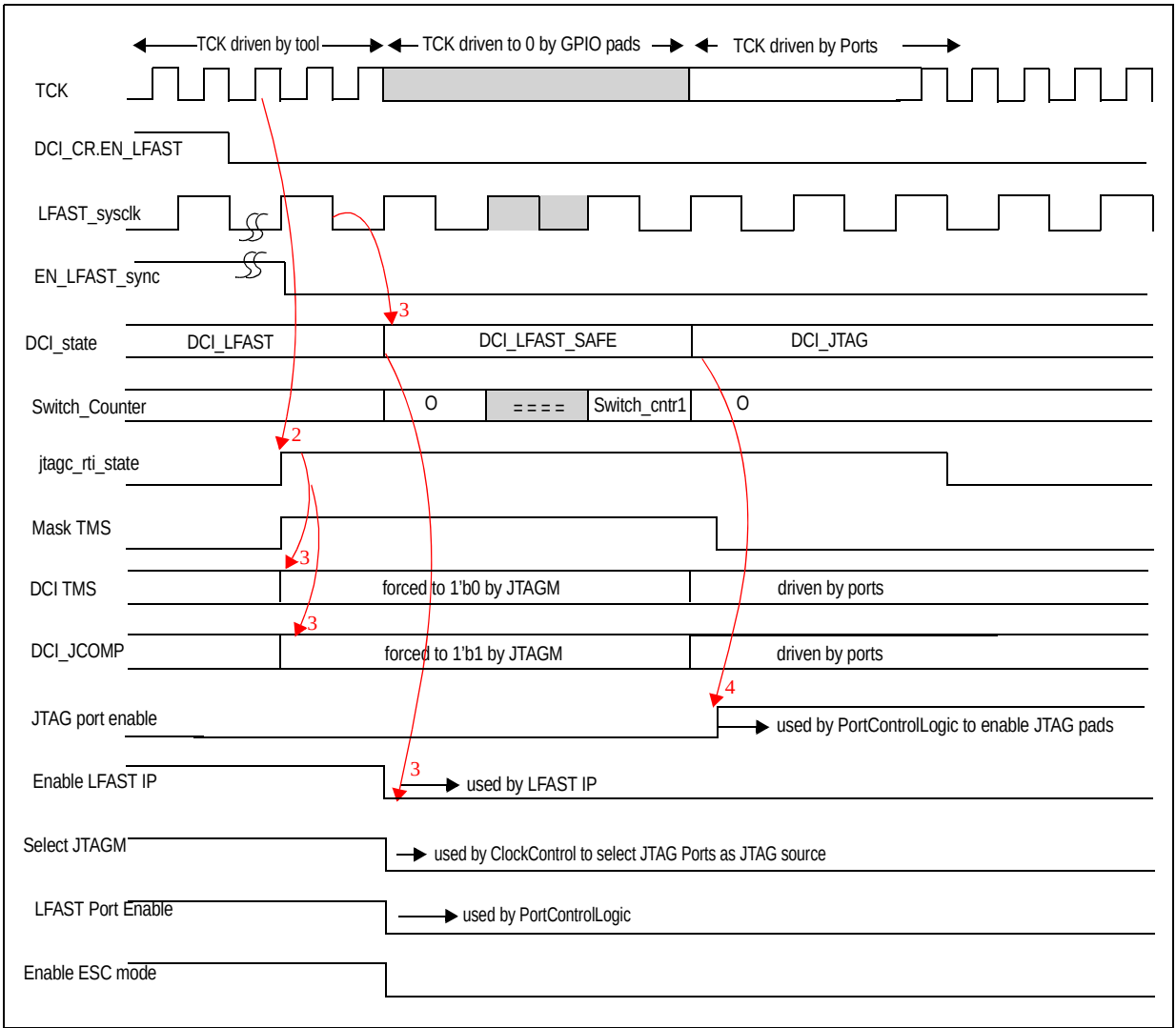


Figure 2176. DCI LFAST to JTAG switching diagram

92.1.3.2 VSS detection

When the LFAST pads are operating in Escape mode, a sensing logic is enabled in the pad which detects and flags any disconnection of the LFAST Rx pads. This flag, if asserted, is used to force the DCI to go to JTAG mode to ensure safe operation. This flag, when asserted, also resets the JTAG TAP to restart in JTAG mode and the EN_LFAST bit is reset accordingly. This flag resets only JTAG TAPs and does not affect any Nexus blocks.

This feature is enabled only when DCI_CR[EN_ESC_MODE] is set. By default this feature is not enabled. To tune the tool behavior it is possible to emulate the pad fault using the DCI_CR[FLVDS_PAD_FLT] bit.

92.1.3.3 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool or LFAST-based tool, it is possible to generate JTAG sequences through software programming using JTAGM IP. The JTAGM, which generates JTAG sequences using LFAST protocol, is also used to generate JTAG sequences using software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

There is a software programmable bit, MCR[jtagm_JCOMP], present in JTAGM block which, when programmed to 0, asserts reset for the JTAGC block, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

Note: .It is recommended to utilize either hardware based debug (JTAG or LFAST) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

If both debug methods are used simultaneously, then an external tool may arbitrarily connect when the SoC has already been halted using the software debug mode. In this case, two consecutive assertions of external JCOMP are necessary to allow the external tool to take control of debug session.

92.1.3.4 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks, with the PD DCI providing the reset signal to all PD debug blocks, and the BD DCI providing the reset signal for all BD debug blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect (applicable to both PD and BD)
- JCOMP (applicable to both PD and BD)
- SoC Debug_Enable signal (applicable to PD only)
- System PLL is out of lock state and not bypassed (applicable to PD only)

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the BD Nexus reset signal remains asserted until the JCOMP signal is asserted. Any subsequent loss of

PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset. The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 2035. Nexus reset truth table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

92.1.3.5 System watchdog control

The BD DCI is also used to enable or disable the Software Watchdog Timer (SWT). The DCI provides a single bit SWT enable signal which is used to control all SWTs in the MCU. For this, the EVTI[0] pin is used. If the EVTI[0] pin remains high when the tool asserts JCOMP, then the SWT is enabled. But if the tool drives the EVTI[0] pin low before asserting JCOMP pin, then the SWT is disabled. By default the SWT is enabled and a tool has an option to enable/disable the SWT using the EVTI[0] pin.

92.1.3.6 Break control

The PD DCI provides central break control among the CPUs and non-CPU debug modules such as GTM, DMA, Watchdog and other peripherals.

There are various different input sources that provide debug requests:

- [BLG_DCI_0044][Covers: ADD23.025][Covers: ADD23.029][Covers: ADD23.134]EVTI[1:0] signals from BD EVTI pads provided to the PD DCI via intra-die pads[end]
- EVTI[1:0] signals from BD JTAGM
- EVTI[1:0] signals from the Sequence Processing Unit (SPU)

The PD DCI generates following debug request signals:

- Debug Request signal to be used for all non-CPU peripherals
- External Debug Request to be used to request external debug event to processor cores
- OnCE Debug Request to be used to request OnCE debug event to processor coresAll break control requests are issued simultaneously to CPUs and peripherals, then synchronized to each clock domain

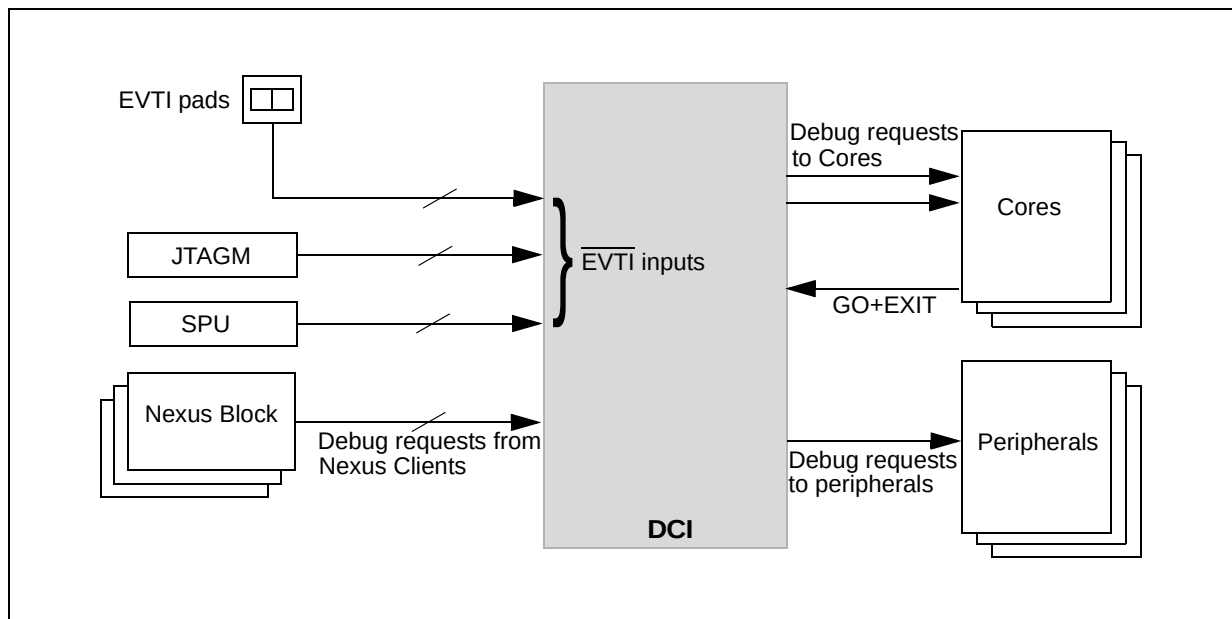


Figure 2177. DCI debug control mechanism

The various sources of $\overline{\text{EVTI}}[1:0]$ inputs are synchronized and logically ANDed in the DCI to generate two debug requests $\text{DCI_EVTI}[1:0]$ for the processor cores and non-CPU peripherals. Of these two inputs, $\text{EVTI}[0]$ is used to generate debug requests for non-CPU peripherals and for processor cores through the external debug request signal. The other input, $\text{EVTI}[1]$, is also used to generate debug requests for non-CPU peripherals and it generates debug requests to processor cores through the OnCE debug request signal.

The external debug request to processor core is asserted as a single cycle pulse and is synchronized on the respective core clock. The OnCE debug request is negated when the acknowledge signal from the Cores is asserted. The CPU(s) remain in the debug state until GO+EX command is executed for that CPU. Exiting a CPU out of debug state by executing GO+EX command does not take other CPUs out of debug state. The peripheral debug request is negated when the processor core(s) execute GO+EXIT command. The $\overline{\text{EVTI}}[1:0]$ signals coming from the pads are enabled for debug control using $\text{DCI_CR}[5:4]$ bits.

In order to selectively enable/disable debug requests to system peripherals, the respective debug enable bit inside the peripheral should be used.

[Figure 2178](#) shows the timing sequence.

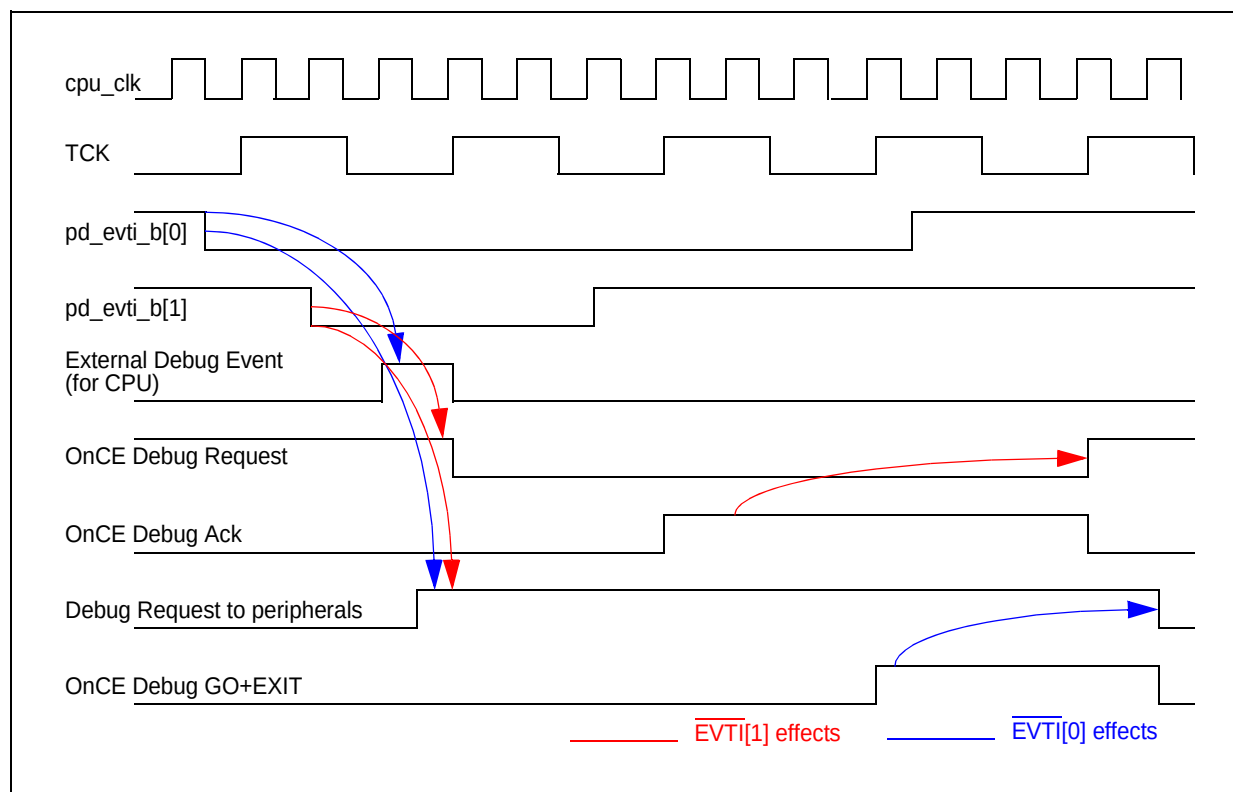


Figure 2178. DCI debug signal generation

92.1.3.7 Cross-triggering control

Many of the blocks in the device, including the Nexus trace blocks and more sophisticated peripherals, have debug features that allow them to request device-wide breaks to enter debug mode. Many blocks can be programmed to halt operation upon debug mode entry or ignore it and continue normal operation. All debug mode entry requests are input to the DCI, ORed together and the result is used to produce the debug requests to the CPUs and non-CPU peripherals.

The cross-triggering feature is implemented through selectively enabling debug control bits in the respective modules. Processor cores (enabled using DBCR0[DEVT1] bit) and non-core peripherals can be programmed to halt operation through their respective control bits. Cross-triggering provides a selective feature to put modules in debug mode upon receiving halt request signals from Nexus modules.

92.1.3.8 Synchronous restart control

The PD DCI implements a synchronous restart control feature to synchronously restart all debug peripherals (CPU and non-CPU peripherals which are put in debug state). This is accomplished with the following steps:

1. The tool sequentially scans into CPUSCR for each processor core to restore the proper processor state.
2. The tool scans into the Production device tap controller the AUX_PARALLEL instruction, which then selects all processor core tap controllers in parallel.
3. To synchronously exit any combination of processor cores from debug mode, the tool simultaneously scans into all processor core tap instruction registers (OCMD) the 10-bit sequence 0x180 (i.e. set GO and EX, clear RS[0:6]).
4. If a processor core is out of debug mode, it sets a flag OnCE GO+EXIT.
5. When the OR of the processor core OnCE GO+EXIT signals is asserted, then all other selected non-processor cores and peripherals in debug would also be synchronously exited from debug mode. It is done by negating peripheral debug request signal.

92.1.3.9 EVTO management

The EVTO management block diagram is presented in [Figure 2179](#).

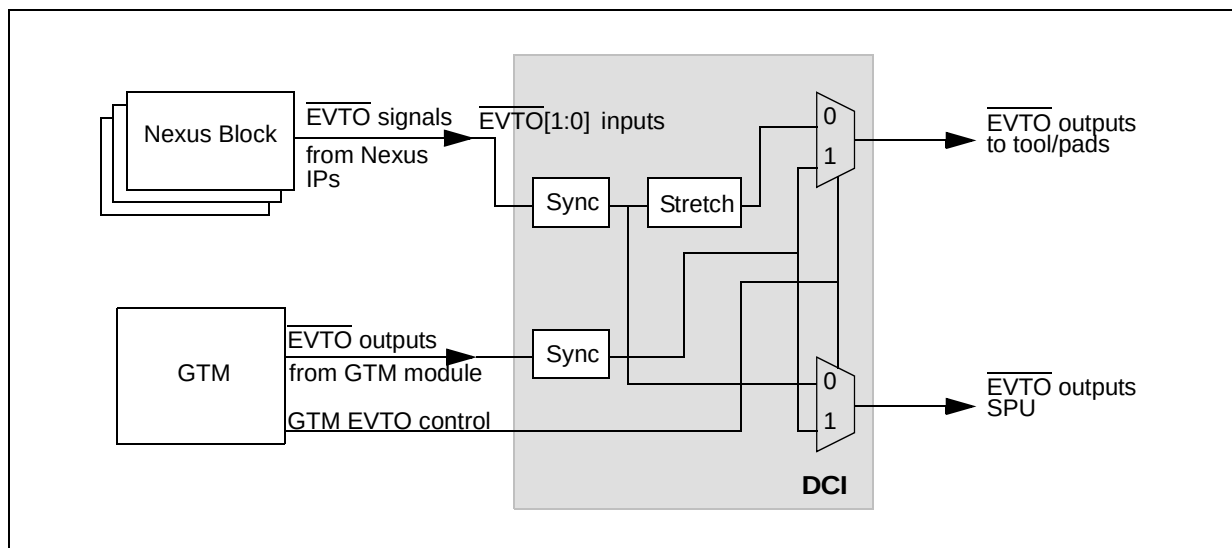


Figure 2179. DCI EVTO management

The PD DCI module controls sharing of the EVTO outputs between all Nexus Clients including the cores that produce EVTO[1:0] signals. The PD DCI uses a high frequency clock to sample EVTO signals coming from all modules which might generate EVTO outputs at their own clock frequency. After receiving a single clock period of asserted EVTO[1:0] signals from any Nexus client, the DCI immediately asserts EVTO[1:0] outputs and stretches it for a minimum of one output clock period for the output to the tool([Figure 2180](#)). The DCI also generates EVTO outputs in other device clock domains so that other Nexus clients can receive EVTO inputs without the need of any synchronization. EVTO function is active as long as the DCI is not in reset.

The PD DCI also allows the GTM to take control of the EVTO outputs by controlling the EOS0/EOS1 fields of the GTMDI_DC register. If these fields are programmed to HALT/TIM/TOM/ATOMEVTO value, the GTM EVTO outputs are directly available on the ports bypassing the stretch logic.

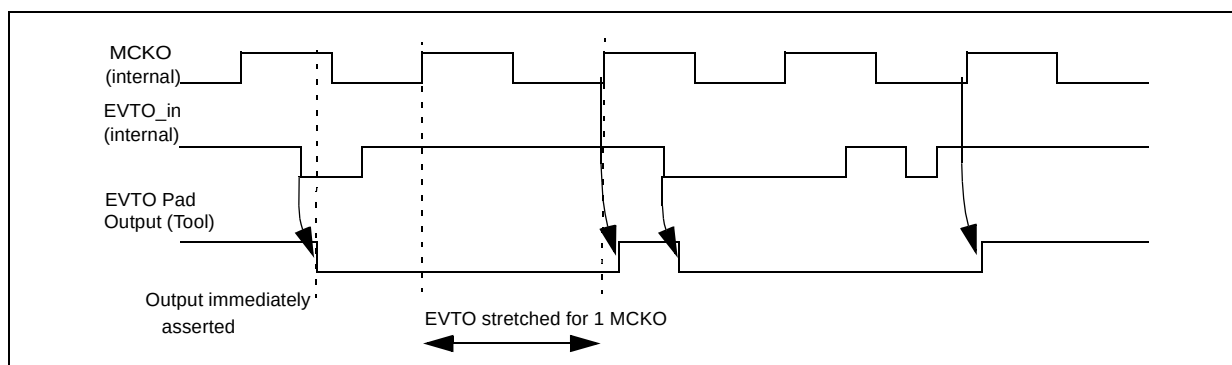


Figure 2180. DCI EVTO timing diagram

92.1.3.10 Reset control

The BD DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- [BLG_DCI_0042][Covers: ADD23.026][Covers: ADD23.032]Force BD Reset—This reset is generated by programming DCI_CR[FBD_RESET] bit. When set it forces reset. This bit has to be reset by the tool to negate the reset signal.[end]

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

92.1.3.11 Security

The SoC security logic provides a Debug_Enable signal to the PD DCI. The PD DCI uses the Debug_Enable signal as a control input to generate debug module reset. When debug disable is requested, the PD DCI keeps all debug blocks in the reset state. This is a protection mode used to disable debug accesses to the peripheral when the device is in a secured state.

92.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

92.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. The BD DCI only contains the DCI_CR and does not contain the DCI_PINCR. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

92.3.1 DCI control register (DCI_CR)

The DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. [Figure 2181](#) shows the format of the DCI_CR.

In an ED which has both PD and BD instances of the DCI, all bits controlling the switch between LFAST and JTAGM must be programmed on the BD DCI instance. The only bits of this register that do not follow this rule are EVTI1_PAD_EN and EVTI0_PAD_EN, which must be programmed in both DCI instances.

Figure 2181. DCI control register (DCI_CR)

	31	30	29	28	27	26	25	24
R	SWITCH_CNTR2							
W								
Reset	1	1	1	1	1	1	1	1
	23	22	21	20	19	18	17	16
R	SWITCH_CNTR2							
W								
Reset	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8
R	SWITCH_CNTR1[4:0]				FLVDS_PAD_FLT	Reserved		
W								
Reset	1	1	1	1	1	0	0	0
	7	6	5	4	3	2	1	0
R	FBD_RES	FORCE_L	EVTI_1_	EVTI_0_	Reserved		EN_ESC_	EN_
W	ET	FAST	PAD_EN	PAD_EN			MODE	LFAST
Reset	0	0	0	0	0	0	0	0

The DCI_CR is described in [Table 2036](#).

Table 2036. DCI_CR field descriptions

Field	Description
31:16 SWITCH_CNTR2	Width of Switch_CNTR2. Decides the second timeout value while switching from JTAG to LFAST mode. The maximum timeout duration = $32,768 \times T_{lfast_sysclk}$
15:11 SWITCH_CNTR1	Width of Switch_CNTR1. Decides the first timeout value while switching from JTAG to LFAST mode, and the timeout value while switching from LFAST mode to JTAG mode. The maximum timeout duration = $32 \times T_{lfast_sysclk}$

Table 2036. DCI_CR field descriptions (continued)

Field	Description
10 FLVDS_PAD_FLT	Force LVDS Pad Fault. Forces the pad fault condition to tune the tool without actually having the pad fault on LFAST Rx pads. This bit is effective only when DCI_CR[EN_ESC_MODE] is set. 0 Do not force Rx LVDS Pad Fault 1 Force Rx LVDS Pad Fault if DCI_CR[EN_ESC_MODE] is set
7 FBD_RESET	Force Reset 0 Do not force Reset 1 Force Reset
6 FORCE_LFAST	Force enable for LFAST 0 LFAST not forced to enable and is controlled using DCI_CR[EN_LFAST] bit 1 LFAST forcefully enabled (without using DCI_CR[EN_LFAST] bit)
5 EVTI1_PAD_EN ⁽¹⁾	$\overline{\text{EVTI}}[1]$ feature enable 0 $\overline{\text{EVTI}}[1]$ disabled (default) 1 $\overline{\text{EVTI}}[1]$ enabled
4 EVTI0_PAD_EN ⁽²⁾	$\overline{\text{EVTI}}[0]$ feature enable 0 $\overline{\text{EVTI}}[0]$ disabled (default) 1 $\overline{\text{EVTI}}[0]$ enabled
1 EN_ESC_MODE	Enable Escape Mode. Fault detect feature enable for LFAST Rx pads 0 Fault detect feature not present for LFAST Rx pads (default) 1 Fault detect feature present for LFAST Rx pads and are enabled after the successful switching to LFAST mode
0 EN_LFAST ⁽²⁾	Enable for LFAST-based tool selection. This bit can be reset either by programming by tool, or pad fault condition in which JTAGC block is reset. 0 LFAST tool not enabled. DCI operates in JTAG mode (default) 1 LFAST tool enabled

1. In an ED that has both PD and BD, EVTI1_PAD_EN and EVTI0_PAD_EN must be programmed in both BD and PD instances of the DCI in order to allow events generated on the BD to pass through the BD DCI and reach the PD DCI.
2. DCI_CR is a JTAG data register and is not accessible for software read. However, the DCI_CR[EN_LFAST] bit is mapped to JTAGM register bit JTAGM_SR[1] to allow for software read access and a mechanism for software to be aware of an active debug tool interface.

92.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter which describes how the modules are configured.

92.4 Functional description

92.4.1 DCI mode transition

After power-on reset, the BD DCI starts in JTAG mode. To enable LFAST mode, the external tool has to first operate in JTAG mode, configure DCI and then switch to LFAST mode. This requires certain careful steps so as to avoid any contention on double-bonded pads for JTAG/LFAST. [Table 2037](#) and [Table 2038](#) show the steps and expected states for the MCU

and tool for JTAG to LFAST switching. [Table 2039](#) and [Table 2040](#) contains the different steps, and the MCU and tool expected states for the LFAST to JTAG switching.

Table 2037. MCU operation (JTAG to LFAST switching)

MCU pin	JCOMP low	JCOMP high (JTAG mode)	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	1st Timeout	2nd Timeout	LFAST ICLC enable command received
TDI	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxN
TMS	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxP
TDO	Hi-Z	Logic Output	Logic Output	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxP
JCOMP	Logic Input	Logic Input	Logic Input Latch	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxN
TCK	Logic Input	Logic Input	Logic Input	Hi-Z	Logic Output	Logic Output	Logic Output
Mode Escape	Inhibit	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable
LFAST_SYSCCLK_EN	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable	Enable

Table 2038. Tool operation (JTAG to LFAST switching)

Tool	JCOMP low	JCOMP high (JTAG mode) ⁽¹⁾	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	TCK detected	Internal timer elapse
TDI	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxN
TMS	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxP
TDO	Hi-Z	Logic Input	Hi-Z	Hi-Z	LVDS Tx	LVDS RxP
JCOMP	Hi-Z	Logic Output (High)	Hi-Z	Hi-Z	LVDS Tx	LVDS RxN
TCK	Don't Care	Logic Output	Logic Input	Logic Input	Logic Input	Logic Input

1. This state is used to set DCI_CR[EN_LFAST] bit.

Table 2039. MCU operation (LFAST to JTAG switching)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
TDI	LVDS Tx	Logic Input	Logic Input
TMS	LVDS Tx	Logic Input	Logic Input
TDO	LVDS Rx	Hi-Z	Logic Output
JCOMP	LVDS Rx	Logic Input Latch	Logic Input (sample JCOMP pin)
TCK	Logic Output	Logic Input	Logic Input

Table 2039. MCU operation (LFAST to JTAG switching) (continued)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
Mode Escape	Enable	Inhibit	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit	Inhibit

Table 2040. Tool operation (LFAST to JTAG switching)

Tool	Send command to clear DCI_CR[EN_LFAST]	LFAST transmits idle after command sent	TCK input clock stops or LVDS Rx both low	Internal timeout: start sending JTAG messages
TDI	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TMS	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TDO	LVDS Tx	Hi-Z	Logic Input	Logic Input
JCOMP	LVDS Tx	Hi-Z	Logic Output (High)	Logic Output (High)
TCK	Logic Input	Logic Input	Logic Output	Logic Output

92.4.2 LFAST LVDS pad fault

It is possible that when the tool is operating in LFAST mode, the tool may get disconnected due to external disturbances, which could put the MCU in an undetermined state. It is imperative to keep the MCU in a safe state. Thus the BD LFAST Rx pads have a sensing logic which detects if the Rx pad voltages are not equal to the common mode voltage when the BD DCI is in LFAST mode. If it is found that the LFAST Rx pads are not corresponding to the common mode voltage, the pad logic sends a signal to the DCI which then puts the DCI into JTAG mode. [Table 2041](#) and [Table 2042](#) contain the different steps and expected states for the MCU and tool for LFAST to JTAGRST switching.

Table 2041. MCU operation (LFAST to JTAGRST switching)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
TDI	LVDS TxN	Logic Input
TMS	LVDS TxP	Logic Input
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Input
TCK	Logic Output	Logic Input
Mode Escape	Enable	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit
Internal TRST	Negated	Asserted

Table 2042. Tool operation (LFAST to JTAGRST switching)

Tool	LFAST mode	Tool forces JCOMP low
TDI	LVDS TxN	Don't Care
TMS	LVDS TxP	Don't Care
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Output (Low)
TCK	Logic Input	Don't Care

93 Buddy Device—JTAG Controller (JTAGC)

93.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

93.1.1 Block Diagram

The following is a block diagram of the JTAG Controller (JTAGC) block.

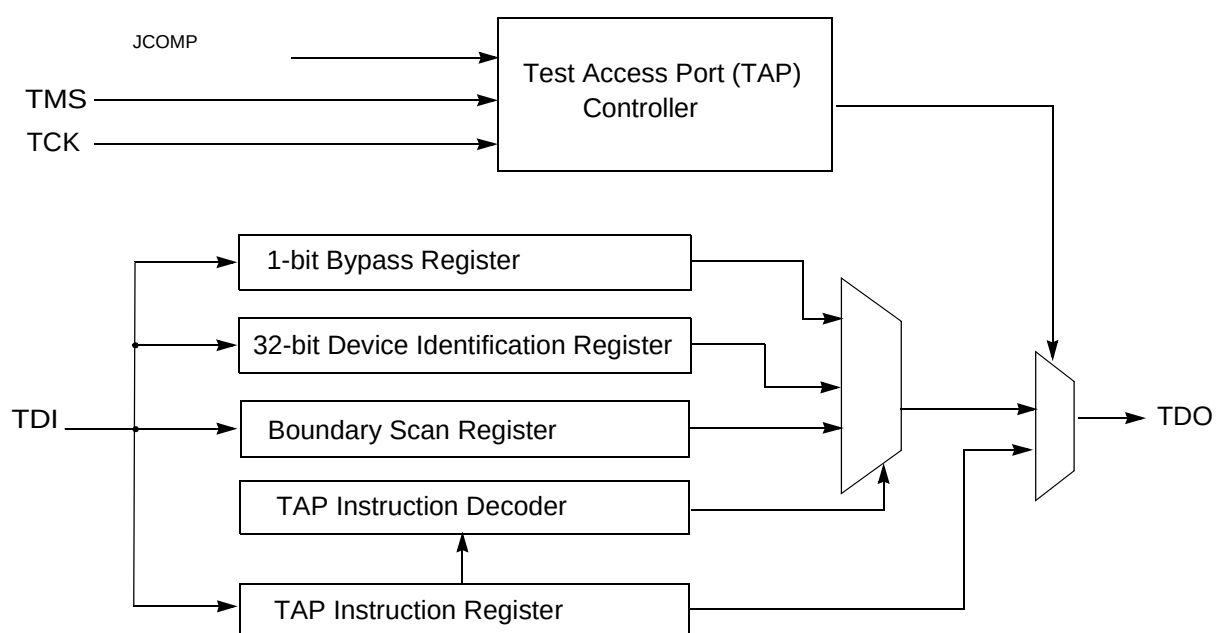


Figure 2182. JTAG (IEEE 1149.1) block diagram

93.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
 - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control and the ability to share the TAP.
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to [Table 2046](#) for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Data registers, bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

93.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

93.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

93.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in [Section 93.4.4, JTAGC block instructions](#).

93.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

93.2 External signal description

The JTAGC consists of five signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 2043. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z ⁽¹⁾	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

93.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

93.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

93.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in [Section 93.4.3, TAP controller state machine](#).

93.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

93.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

93.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

93.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	5	4	3	2	1	0
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

Figure 2183. 6-bit instruction register

93.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

93.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

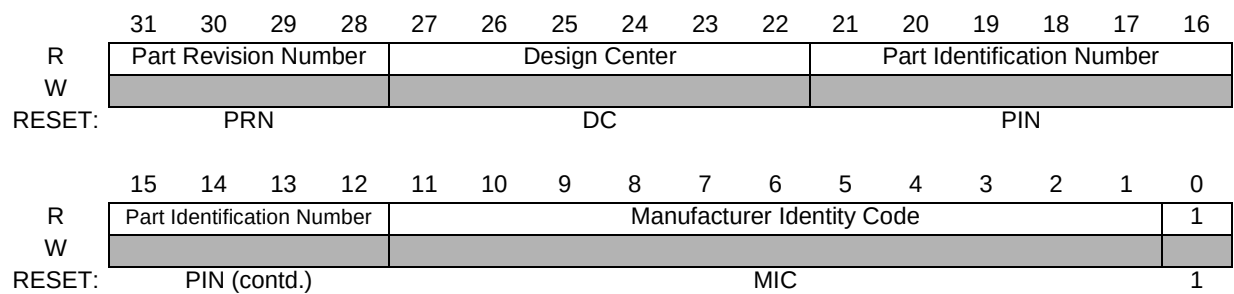


Figure 2184. Device identification register

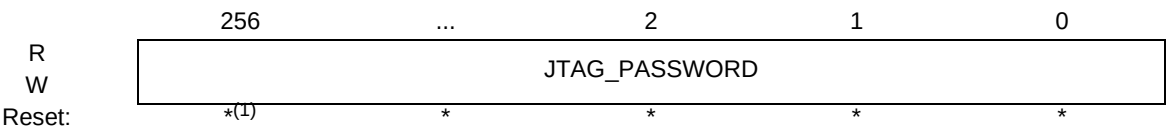
The following table describes the device identification register functions.

Table 2044. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x0.
DC	Design Center. Indicates the design center. Value is 0x2B.
PIN	Part Identification Number. Contains the part number of the device. Value is 0x381.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x020.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

93.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256'b0.

Figure 2185. JTAG_PASSWORD register

The following table describes the JTAG_PASSWORD register functions.

Table 2045. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

93.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in [Section 93.4.5, Boundary scan](#). The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

The BD boundary scan chain includes registers connected to all PD/BD interface signals at the BD side. These registers are controlled by the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions applied to the BD JTAG controller. Similarly, the PD boundary scan chain also contains registers connected to the interface signals at the PD side, which are controlled by instructions applied to the PD JTAG controller.

93.4 Functional description

This section explains the JTAGC functional description.

93.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

93.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to [Section 93.4.4.8, ACCESS_AUX_x instructions](#).

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

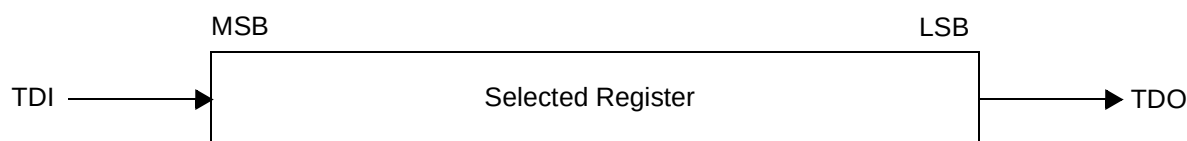
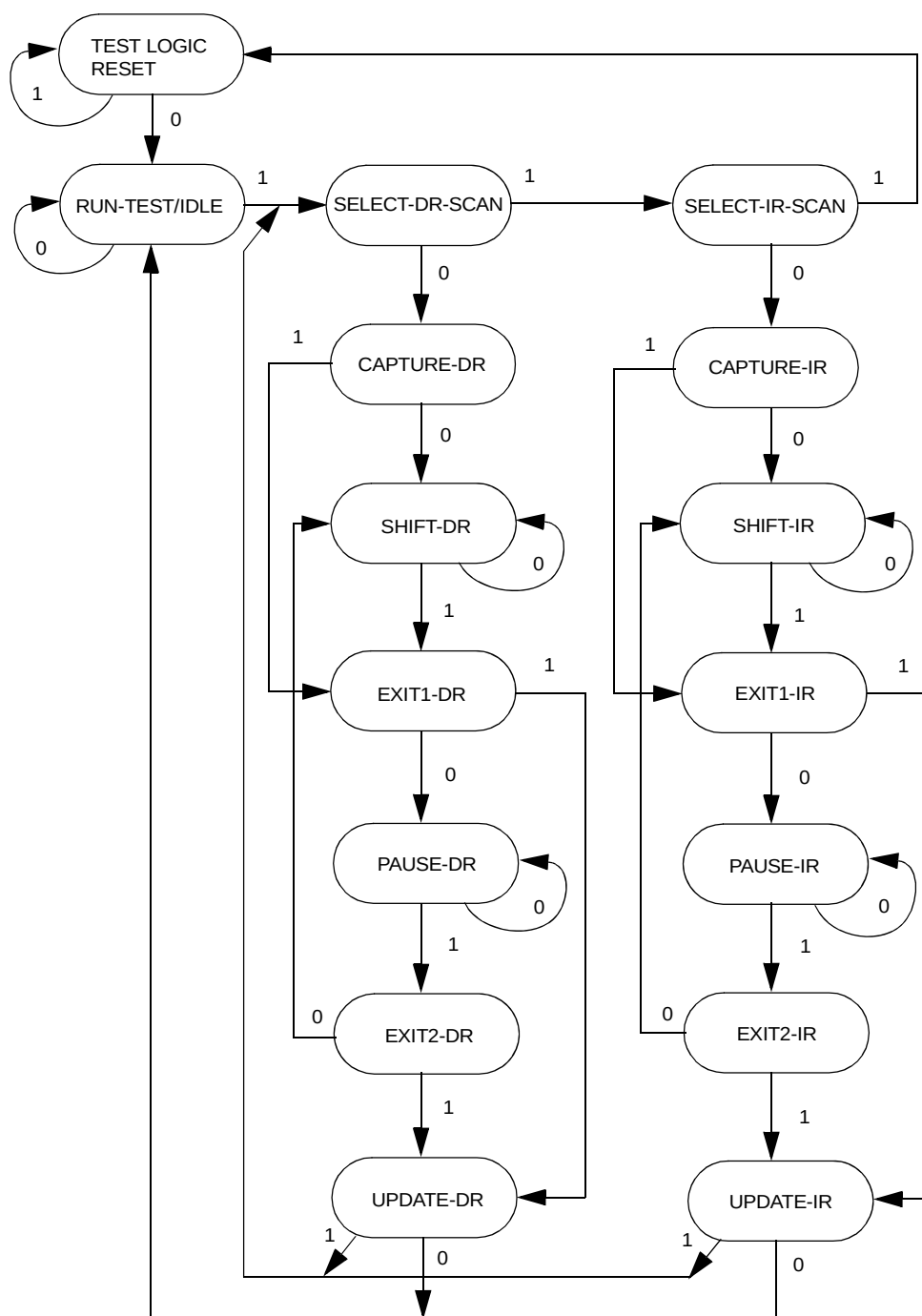


Figure 2186. Shifting data through a register

93.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition represents the value of TMS at the time of a rising edge of TCK.

Figure 2187. IEEE 1149.1-2001 TAP controller finite state machine

93.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

93.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

93.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 2046. 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
Factory debug reserved	000101	Intended for factory debug only
Factory debug reserved	000110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register while three-stating all output pins and asserting functional reset
Factory debug reserved	001010	Intended for factory debug only
CLAMP	001100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ENABLE_DCI_CR	001110	Enables access to DCI CR register
Reserved	001111	Reserved
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
ACCESS_AUX_NR_BD	100000	Enables access to the NAR TAP controller on BD
Reserved	100001	Reserved

Table 2046. 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
ACCESS_AUX_NAL	100010	Enables access to the NAL TAP controller
ACCESS_AUX_RWA_BD	100011	Enables access to the RWA TAP controller
Reserved	100100	Reserved
Factory test reserved	100101	Intended for factory test only
Reserved	100110	Reserved
Reserved	100111	Reserved
Reserved	101000	Reserved
Reserved	101001	Reserved
Reserved	101010	Reserved
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
Reserved	110011	Reserved
Reserved	110100	Reserved
Reserved	110101	Reserved
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
ACCESS_AUX_DWPU	111001	Reserved
Reserved	111010	Reserved
Reserved	111011	Reserved
Reserved	111100	Reserved
Reserved	111101	Reserved
ACCESS_AUX_DCI_PD	111110	Enables access to pass control from BD to PD
BYPASS	111111	Selects bypass register for data operations
Reserved ⁽¹⁾	All other opcodes	Decoded to select bypass register

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

93.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

93.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- First, the SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- Secondly, the PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

93.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

93.4.4.4 EXTEST—External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

93.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

93.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

93.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

93.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

93.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

93.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

PD and the BD have their own separate boundary scan chains, each one controlled by the respective JTAG controller. Besides the package I/O pins, these chains also encompasses the interface signals between BD and PD. Testing of the interface can be done by applying EXTEST, SAMPLE or SAMPLE/PRELOAD instructions to both PD and BD JTAG.

93.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any

mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

94 Buddy Device—JTAG Master (JTAGM)

94.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

94.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface or from the LFAST through the parallel interface. The JTAGM has the capability to differentiate between data from software and LFAST. It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI and transfers this data to the LFAST through another parallel interface. The JTAGM has the following registers:

- Configuration register
- Status register
- Four data output registers
- RxCRC receive register
- Two data input registers

All these registers, described in detail in [Section 94.4, Memory map and register definition](#), are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

[Figure 2188](#) shows the block diagram of the JTAGM.

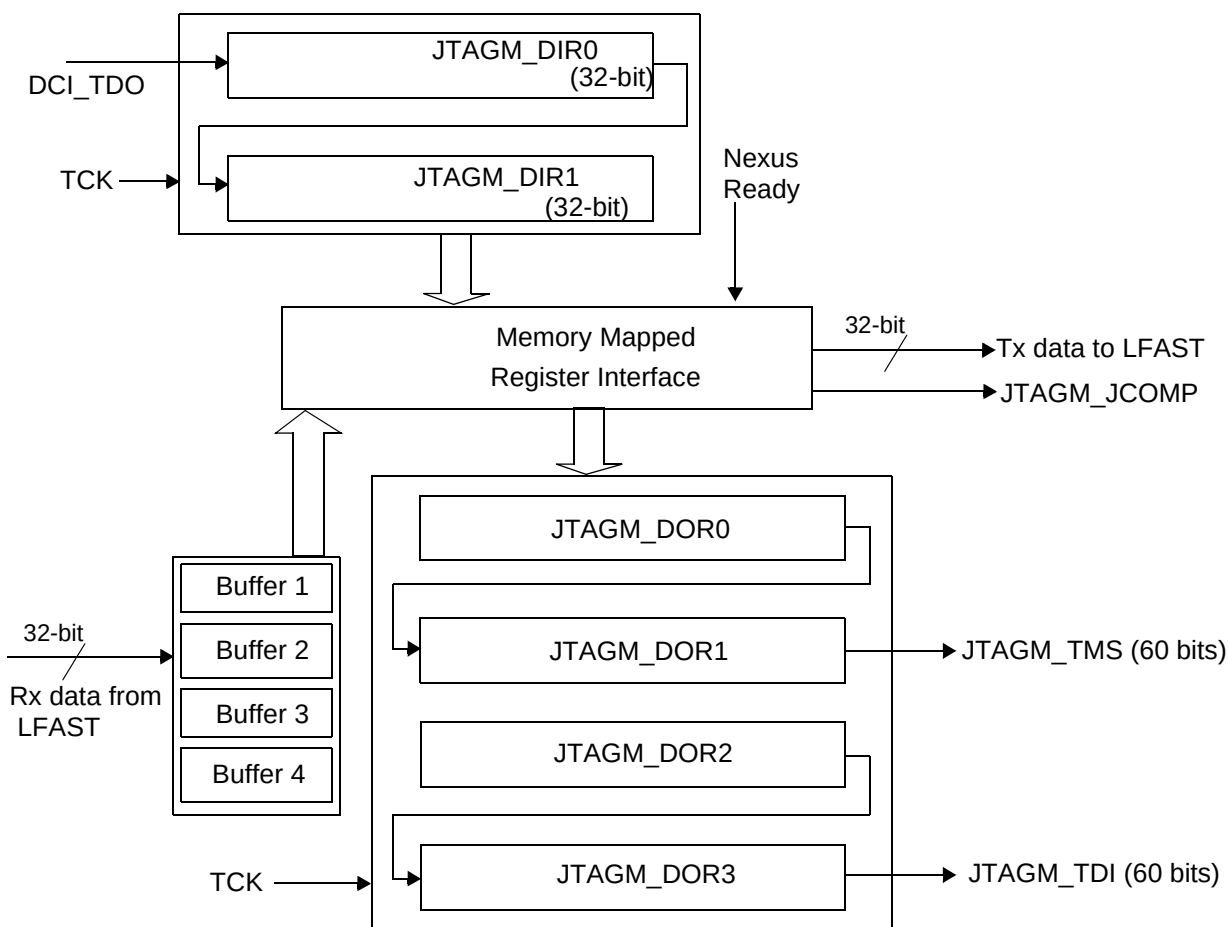


Figure 2188. JTAGM block diagram

94.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG
- Receives/provides JTAG data from/to the LFAST

94.2 Functional description

94.2.1 Module architecture

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and

capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

Figure 2189 shows the flow diagram of the LFAST to DCI connection.

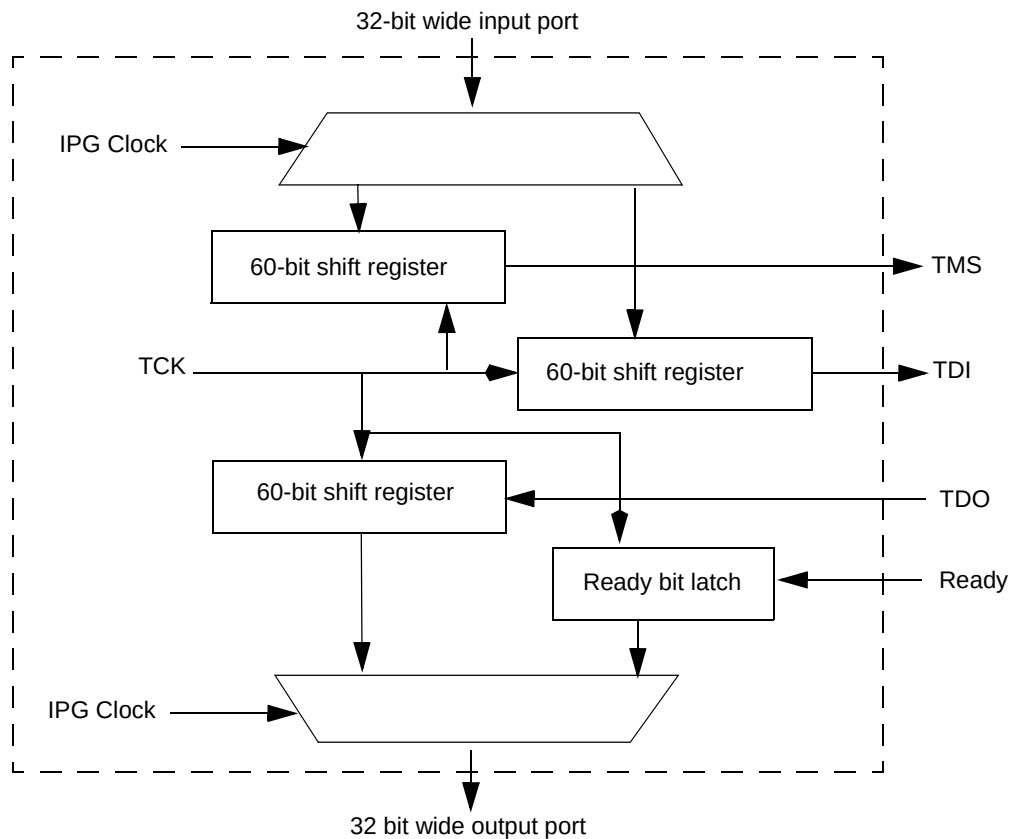


Figure 2189. LFAST-to-DCI flow diagram

94.2.2 JTAGM JCOMP usage in LFAST

JTAGM is used to drive JTAG sequences in software mode and in LFAST mode. The JTAGM_MCR[jtagm_JCOMP] bit is provided to reset all TAP controllers by generating a JCOMP signal to JTAGC.

The guidelines to be considered for the usage of JTAGM_MCR[jtagm_JCOMP]:

- Do not clear the JTAM_MCR[jtagm_JCOMP] bit while in LFAST mode as this operation is not guaranteed
- If for some reasons during the LFAST mode there is a loss of communication, one of the following should be done (instead of clearing the jtagm_JCOMP bit):
 - Generate the escape sequence and then, once the device is back in JTAG mode, generate a test-logic-reset state via the TCK and TMS signals. This resets all TAP controllers but does not affect the client debug configuration
 - Generate the escape sequence and then, once the device is back in JTAG mode, force the JCOMP pin low. This resets all TAP controllers as well as the client debug configuration

Note: The escape sequence is generated by forcing LFAST LVDS ports RX+ and RX- to zero.

94.2.3 JTAGM data error detection

To improve the integrity of the JTAG operation, data passed to and from the JTAGM includes an 8-bit CRC only in LFAST mode. The data sent to the JTAGM consists of 60 bits of TMS data, 60 bits of TDI and an 8-bit CRC. The JTAGM recalculates the 8-bit CRC for the first 120 bits of data and compares the result with the received CRC. If the two match, the data is assumed to be valid and the JTAGM converts this data into a valid JTAG frame.

If the CRCs do not match, the module sets the CRC error status bit (CRC_err) in the JTAGM status register (JTAGM_SR) and sends a standard 96-bit response frame to the LFAST channel B, with all 60 data bits set to 0. The CRC error status bit is automatically cleared as soon as the response frame has been sent to the LFAST.

Data sent from the JTAGM to the LFAST also includes an 8-bit CRC. The data sent from the JTAGM is in the form of a 96-bit frame consisting of 60 bits of TDO data + 4 bits of zero-padding + 32 bits of the status register. The 8-bit CRC value is calculated for the first 88 bits of the 96-bit frame and inserted into the least significant 8 bits of the status register; which is also the last 8 bits of the 96-bit frame. The tool is responsible for checking the CRC of the received data and taking corrective action.

94.2.4 JTAGM message tagging

The JTAGM, in conjunction with the LFAST, is pipelined to allow several JTAGM messages to be sent from the tool before any response is received back. This allows better utilization of the LFAST bandwidth. To enable the tool to associate individual messages with the correct response, the messages are tagged. The tool sends data messages on LFAST channels E, F, G and H in sequence. The JTAGM module returns responses on the same channel they were received. For example, if the tool sends a data command on channel E, JTAGM returns the response on channel E; if the tool sends a data command on channel F, the JTAGM returns the response on channel F; and so on. This way, the tool can ensure a response was generated by a specific command.

Configuration commands to write/read the configuration and status registers within the JTAGM are sent on channel A and response data is also sent back on channel A.

94.2.5 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available. In conventional JTAG mode, the status of this ready bit is also provided back to the tool via a dedicated pin to allow the tool to know data is ready and the JTAG read register can be accessed. With the LFAST interface providing pipelined data to the JTAGM, this technique is not available. Two interlocked mechanisms are provided within the JTAGM that use this ready bit to allow optimized data reads without ready bit errors.

94.2.5.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK

shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

94.2.5.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

94.2.6 EVTO and EVTI signals

The standard Nexus EVTI (Event In) and EVTO (Event Out) signals are optionally present on dedicated package pins and are used by an external tool in conjunction with the JTAG interface to pass information, events and triggers between the tool and the MCU.

The JTAGM as well as providing a higher speed alternative to the standard JTAG interface, also allows control and status of the EVTI and EVTO signals to be embedded into the serial protocol. The EVTIO and EVTI1 signals can be asserted by writing control bits in the Module Configuration Register (JTAGM_MCR). The JTAGM can be configured by bits in the Module Configuration Register (JTAGM_MCR) to respond to a rising/falling/both edge transition of either EVTO0 or EVTO1. The response is either an unsolicited serial message to the tool when in LFAST mode (DTM = 0); or a CPU interrupt when in software mode (DTM = 1).

The unsolicited LFAST message is sent on LFAST channel B to allow the tool to distinguish the message from normal JTAG response data (channels E, F, G & H) and Config Command Responses (channel A). The CPU interrupt is gated by an Interrupt enable bit, in MCR and reported by Interrupt Status flags in the Status Register (JTAGM_SR). The status bits stay set until cleared by writing to the clear bits in the JTAGM_SR.

Asserting the evti0_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. Asserting the evti1_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. The DTM bit in the JTAGM_MCR register can put the DCI in LFAST or SW mode.

94.2.7 JTAGM configuration and status monitoring

The JTAGM module requires some level of configuration and status monitoring. The JTAGM includes a 32-bit configuration register (JTAGM_MCR) and a 32-bit status register (JTAGM_SR, which includes the ready bit mentioned in [Section 94.2.5.1, Status register ready bit](#)). The configuration register is accessed via the parallel input port by a 128-bit

payload arriving on channel A. The 128-bit payload is made from the following:

- 32-bit mask of which bits within the configuration register should be written +
- 32-bit value to be written to the configuration register +
- 24-bit mask of which bits within the status register should be written +
- 8 bits of zero-padding +
- 24-bits of data to be written to the status register +
- 8-bit CRC (The least significant 8 bits of the status register cannot be written because they contain automatically generated CRC data).

The masked data is transferred to the 32-bit configuration register and 32-bit status register. The JTAGM module responds to this configuration/status write by outputting a 96-bit payload to the parallel output port to channel number A. The 96-bit payload is made up of the 32-bit value of the configuration register plus 32 bits of zero-padding plus 32-bit value of the status register (including the least significant 8 bits which are an 8-bit CRC calculated on the previous 88 bits).

While the current messages are being transmitted from the second buffering (memory mapped registers), the first buffers are continuously filled up. The status register also includes status bits provided by the LFAST module and the DCI module. The JTAGM requests data from the LFAST only when at least one internal buffer is free.

94.2.8 JTAGM to DCI serial interface

Figure 2190 shows the data transfer timings between JTAGM and DCI.

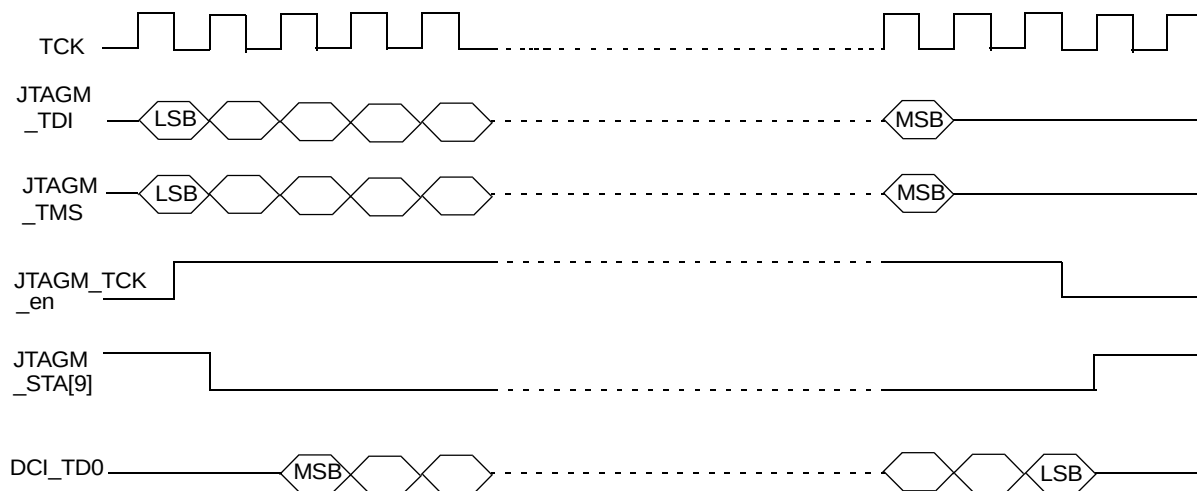


Figure 2190. JTAGM to DCI serial interface diagram

94.2.9 JTAGM data handling and CRC calculation in LFAST mode

Figure 2191 shows the bit ordering when a data frame is transmitted from LFAST to JTAGM for subsequent transmission to DCI.

Figure 2191. Bit ordering for LFAST to JTAGM transfer

bit 127 to bit 68				bit 67 to bit 8				7	6	5	4	3	2	1	0
TMS (60 bits)				TDI (60 bits)				CRC (8 bits)							
bit 0 to bit 59				bit 0 to bit 59											
Last TMS bit transmitted by JTAGM		First TMS bit transmitted by JTAGM		Last TDI bit transmitted by JTAGM		First TDI bit transmitted by JTAGM									

[Figure 2192](#) shows the bit ordering when a data frame is transmitted from JTAGM to LFAST consisting of TDO data received from DCI.

Figure 2192. Bit ordering for JTAGM to LFAST transfer

bit 95 to bit 36			bit 35 to bit 32			bit 31 to bit 8			7	6	5	4	3	2	1	0
TDO (60 bits)			0 pad (4 bits)			JTAGM_SR (24 bits)			CRC (8 bits)							
bit 0 to bit 59			bit 0 to bit 59			bit 31 to bit 8										
First TDO bit received by JTAGM		Last TDO bit received by JTAGM														

The CRC calculation is described in following example.

The data transferred from LFAST to JTAGM (in four frames) is as follows:

- D0—F7B3_D591
- D1—E6A2_C486
- D2—A2C4_80F7
- D3—B3D5_9183

These are supplied to CRC block starting from D0 to D3. The CRC calculation begins on D0 with the MSB (left bit) going first. D3 is supplied as B3D5_9100 (after masking CRC).

JTAGM extracts TMS and TDI information from the data received as follows:

- TMS = F7B3_D591_E6A2_C48
- TDI = 6A2C_480F_7B3D_591

JTAGM transmits the data to DCI starting from LSB (right bit going first). Corresponding to this TMS and TDI transmission, JTAGM receives TDO data from DCI and stores it as first bit to most significant location in TDO frame (shown in [Figure 2192](#)).

94.2.10 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

94.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

94.4 Memory map and register definition

[Table 2047](#) contains the memory map of the JTAGM registers.

Table 2047. Memory map

Offset	Register	Location
0x00	Module Configuration Register (JTAGM_MCR)	on page 3785
0x04	Status Register (JTAGM_SR)	on page 3787
0x08	Data Out Register 0 (JTAGM_DOR0)	on page 3790
0x0C	Data Out Register 1 (JTAGM_DOR1)	on page 3790
0x10	Data Out Register 2 (JTAGM_DOR2)	on page 3791
0x14	Data Out Register 3 (JTAGM_DOR3)	on page 3791
0x18	Receive CRC Register (JTAGM_RxCRC)	on page 3792
0x1C	Data Input Register 0 (JTAGM_DIR0)	on page 3792
0x20	Data Input Register 1 (JTAGM_DIR1)	on page 3793

94.4.1 Register descriptions

94.4.1.1 Module configuration register (JTAGM_MCR)

This register contains the status bits for the current transfer. [Figure 2193](#) shows the format of the JTAGM_MCR. In the Emulation and Debug Device (ED), all bits of this register need to be programmed in the Buddy Device instance of the JTAGM. The only exception is the SIE bit, which always needs to be programmed in the Production Device instance of the JTAGM, regardless of being ED or not.

Figure 2193. JTAGM_MCR register

Offset 0x00																Access: User read/write																		
	0				1		2		3		4		5		6		7		8		9		10		11		12		13		14		15	
R	SWRESET																		0		0		0		0		0		0		0		0	
W																																		
	evto0_sense				evto1_sense				evto_I		E		evti0_assert		evti1_assert																			
Reset	0				0		0		0		0		0		1		1		0		0		0		0		0		0		0		0	
	16				17		18		19		20		21		22		23		24		25		26		27		28		29		30		31	
R	0				0		inter_jtag_frame_timer										0		SIE		IIE		TCKSEL		jtag_JCOMP		DTM							
W																																		
Reset	0				0		0		0		0		0		0		0		0		0		0		0		0		0		1		0	

The JTAGM_MCR register is described in [Table 2048](#).

Table 2048. JTAGM_MCR register field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit. Writing a 0 has no effect. <i>Note: After being written to "1", this bit is auto cleared within 3 TCK cycles.</i>
1:2 evto0_sense	Determines JTAGM response to activity on $\overline{\text{EVTO0}}$ 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO0}}$ edges (default setting) 01 evto0_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO0}}$ 10 evto0_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO0}}$ 11 evto0_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO0}}$
3:4 evto1_sense	Determines JTAGM response to activity on $\overline{\text{EVTO1}}$ line 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO1}}$ edges (default setting) 01 evto1_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO1}}$ 10 evto1_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO1}}$ 11 evto1_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO1}}$
5 evto_IE	Enables $\overline{\text{EVTO}}$ triggered JTAGM interrupt 0 No JTAGM interrupt is generated In response to any $\overline{\text{EVTO0}}$ or $\overline{\text{EVTO1}}$ activity 1 JTAGM interrupt is generated when evto0_edge or evto1_edge are set
6 evti0_assert	EVTI0 Assert. 0 Set jtagm_evti0 low/asserted 1 Set jtagm_evti0 high/de-asserted
7 evti1_assert	EVTI1 Assert. 0 Set jtagm_evti1 low/asserted 1 Set jtagm_evti1 high/de-asserted <i>Note: In the SW mode(when DTM =1) , regardless of the state of "evti1_assert" bit the internall "evti" signal remains de-asserted . Though this bit can be written , yet writing to this bit has no effect .</i>
8:17	Reserved
18:23 inter_jtag_frame_timer	TCK delay. 0000000 TCK delay 0000011 TCK delay ... 11111163 TCK delay
24	Reserved
25 SIE	SPU Interrupt Enable. ⁽¹⁾ 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer

Table 2048. JTAGM_MCR register field descriptions (continued)

Field	Description
27:29 TCKSEL	TCK clock frequency selection. 000 Not to be used 001 TCK is system clock ÷ 2 010 TCK is system clock ÷ 3 ... 111 TCK is system clock ÷ 8 <i>Note: When using the JTAGM, this bit should always be programmed with any value other than "000".</i>
30 jtagm_JCOMP	JTAG reset. 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data for JTAG transferred by LFAST 1 Data for JTAG transferred by software

1. The BD JTAGM instance does not receive signals from the SPU. If required, this feature needs to be enabled in the PD instance of the JTAGM, even when JTAG generation in LFAST mode is handled by the BD JTAGM.

94.4.1.2 Status register (JTAGM_SR)

94.4.1.2.1 LFAST mode

[Figure 2194](#) shows the format of the JTAGM_SR in LFAST mode. In the ED, all bits of this register need to be accessed in the BD instance of the JTAGM. The exceptions are the bits related to the SPU interrupts (SPU_INT, SPU_INT_CLR), which always need to be accessed from the PD JTAGM, regardless of being ED or not.

Figure 2194. JTAGM_SR register in LFAST mode

Offset 0x04																Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	overrun	0	lfast_status								dci_status																				
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_edge	evto1_edge	SPU_INT_CLR	SPU_INT	CRC_err	Nexus_err	Idle	NR	CRC							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

94.4.1.2.2 SW mode

Figure 2195 shows the format of the JTAGM_SR in SW mode.

Figure 2195. JTAGM_SR register in SW mode

Offset 0x04									Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15								
R	0	0	lfast_status						dci_status															
W																								
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0								

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_edge	evto1_edge	SPU_INT_CLR	SPU_INT	CRC_err	Nexus_err	Idle	NR			0	0	0	0	0	0
W										evto0_clr	evto1_clr					
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

The JTAGM_SR register is described in Table 2049.

Table 2049. JTAGM_SR register field descriptions

Field	Description
0 overrun	Set by hardware when the JTAGM overwrites the first Tx data with the second Tx data. Writing 1 clears this bit. This bit is set only in LFAST mode.
1	Reserved
2:7 lfast_status	Status information from the LFAST to be used by software. The composition of these bits are as follows: bit 2 Indicates either data frame or ping frame was transmitted successfully bit 3 Indicates Tx data Interface not enabled and a frame is ready to be transmitted bit 4 Indicates Rx data FIFO overflow bit 5 Indicates reception of frame with invalid frame payload size bit 6 Indicates reception of frame with invalid ICLC code bit 7 Indicates reception of illegal LCT frame
8:15 dci_status	Status information from the DCI. The composition of these bits are as follows: bit 8 Reserved to 0 bit 9 DCI LVDS port enable bit 10 DCI JTAG port enable bit 11 DCI LVDS safe mode bit 12 DCI JTAG safe mode bit 13 DCI LVDS escape mode bit 14 DCI enable LFAST bit 15 DCI tool present , set when a debug tool is connected

Table 2049. JTAGM_SR register field descriptions (continued)

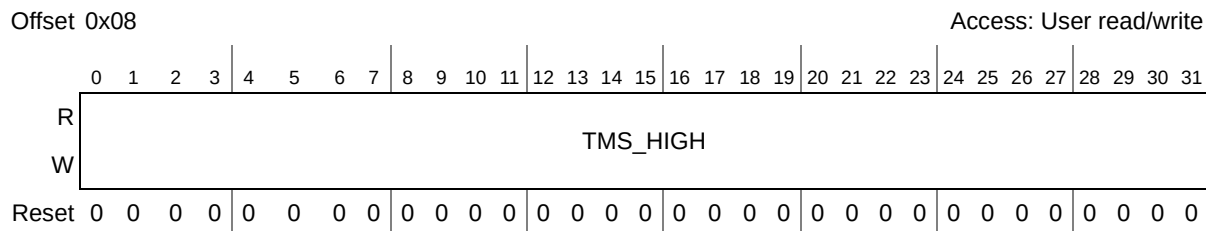
Field	Description
16 evto0_edge	In LFAST mode (DTM = 0) a channel B message is sent on LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit 0 No activity sensed on $\overline{\text{EVTO0}}$ 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO0}}$
17 evto1_edge	In LFAST mode (DTM = 0) a channel B message is sent to the LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto1_clr bit 0 No activity sensed on $\overline{\text{EVTO1}}$ 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO1}}$
18 SPU_INT_CLR	Write 1 clears the SPU_INT ⁽¹⁾ This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit. ⁽¹⁾
20 CRC_err	CRC error bit. Set when the calculated CRC does not match the last received CRC. This bit is cleared when the error packet is sent to the LFAST on channel B.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to 1 when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read <i>Note: In LFAST mode (when DTM = 0), this bit has no effect and acts as a "don't care". This bit should not be used by the application in the LFAST mode</i>
24:31 CRC	Calculated CRC. CRC calculated on the information received. LFAST Mode: Bit 24:31 reads the CRC. Write has no effect. SW Mode: – Bit 24:31 reads as 0x00. Writing a 0 to any bit have no effect. – Writing 1 on bit 24 clears evto0_edge – Writing 1 on bit 25 clears evto1_edge

1. The BD JTAGM instance does not receive signals from the SPU. If required, this bit needs to be read from the PD instance of the JTAGM, even when JTAG generation in LFAST mode is handled by the BD JTAGM.

94.4.1.3 Data out register 0 (JTAGM_DOR0)

[Figure 2196](#) shows the format of the JTAGM_DOR0.

Figure 2196. JTAGM_DOR0 register



The JTAGM_DOR0 register is described in [Table 2050](#).

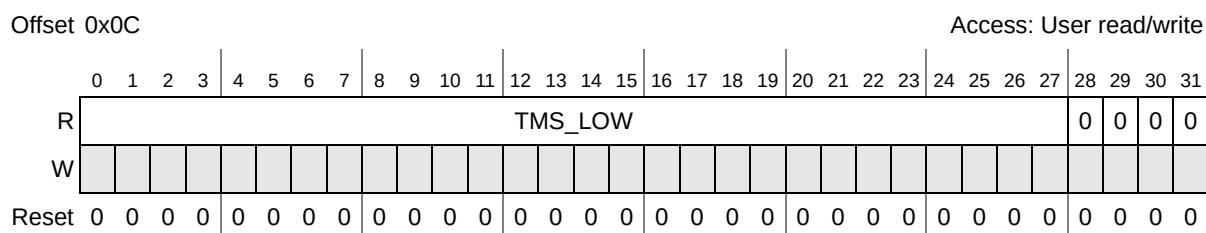
Table 2050. JTAGM_DOR0 register field descriptions

Field	Description
0:31 TMS_HIGH	Higher word of data for TMS, bits 59–28. Writable by software only when DTM = 1

94.4.1.4 Data out register 1 (JTAGM_DOR1)

[Figure 2197](#) shows the format of the JTAGM_DOR1.

Figure 2197. JTAGM_DOR1 register



The JTAGM_DOR1 register is described in [Table 2051](#).

Table 2051. JTAGM_DOR1 register field descriptions

Field	Description
0:27 TMS_LOW	Lower word of data for TMS, bits 27–0. Writable by software only when DTM = 1. TMS_LOW[0] is shifted out first
28:31	Reserved

Figure 2198 shows the format of the JTJGM DOR2.

[illegible]

Table 2052. JTAGM DOR2 register field descriptions

Figure 2199 shows the format of the JTAGM DOR3.

[illegible]

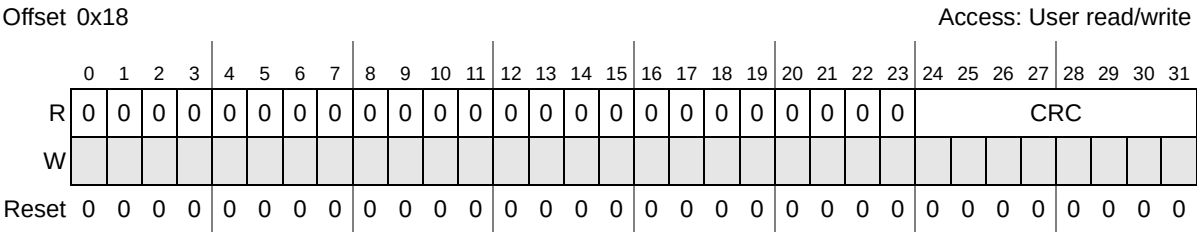
Table 2053. JTAGM_DOR3 register field descriptions

Field	Description
0:27 TDI_LOW	Lower word of data for TDI, bits 27–0. Writable by software only when DTM = 1. TDI_LOW[0] is shifted out first
28:30	Reserved
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

94.4.1.7 Receive CRC register (JTAGM_RxCRC)

Figure 2200 shows the format of the JTAGM_RxCRC register.

Figure 2200. JTAGM_RxCRC register



The JTAGM_RxCRC register is described in Table 2054.

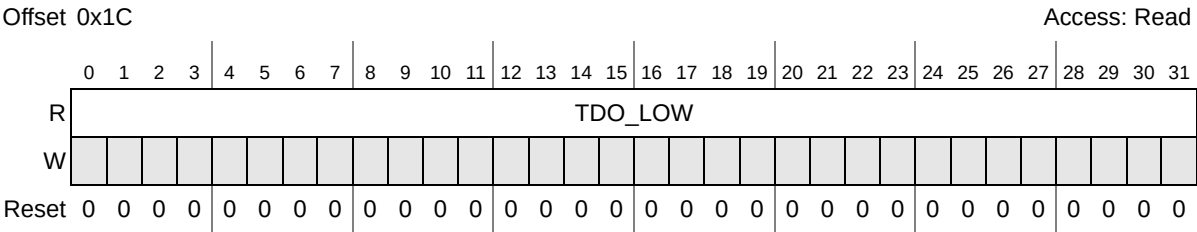
Table 2054. JTAGM_RxCRC register field descriptions

Field	Description
0:23	Reserved
24:31 CRC	Received CRC. Refers to the CRC of the received LFAST message.

94.4.1.8 Data input register 0 (JTAGM_DIR0)

Figure 2201 shows the format of the JTAGM_DIR0.

Figure 2201. JTAGM_DIR0 register



The JTAGM_DIR0 register is described in Table 2055.

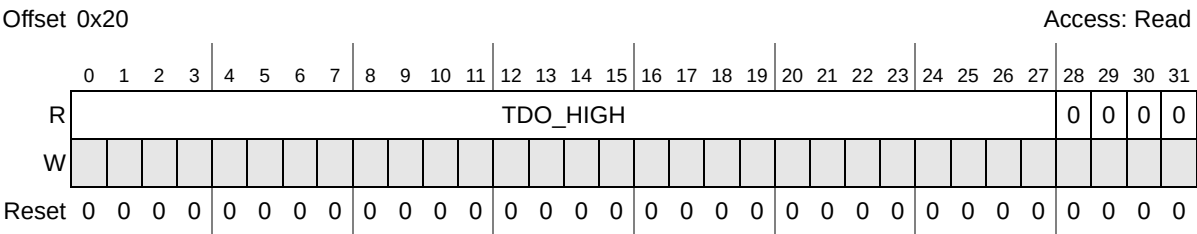
Table 2055. JTAGM_DIR0 register field descriptions

Field	Description
0:31 TDO_LOW	Lower word of data received on TDO, bits 0–31

94.4.1.9 Data input register 1 (JTAGM_DIR1)

Figure 2202 shows the format of the JTAGM_DIR1.

Figure 2202. JTAGM_DIR1 register



The JTAGM_DIR1 register is described in Table 2056.

Table 2056. JTAGM_DIR1 register field descriptions

Field	Description
0:27 TDO_HIGH	Higher word of data received on TDO, bits 32–59
28:31	Reserved

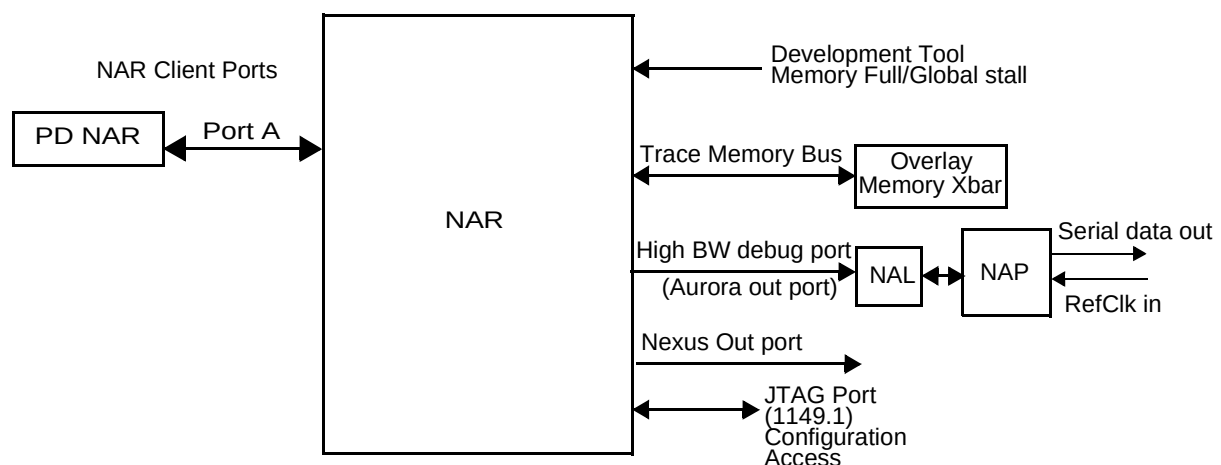
95 Buddy Device—Nexus Aurora Router (NAR)

95.1 Introduction

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

Top-level architecture is shown in [Figure 2203](#). Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured. The Nexus Aurora Link (NAL) takes the 32-bit Nexus message beats from the NAR and processes them in Aurora-ready data octets for the Nexus Aurora Phy (NAP) block, which interfaces directly with the high-bandwidth port. The Aurora output port of NAR can be fed as a client to another NAR.

Figure 2203. NAR top level block diagram



The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. The Nexus Out port provides additional flexibility to the channel trace data to memory on the Nexus recipient. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. These two partitions of the main queue are supported for better throughput each having 50% size. The two NAR output ports can be in operation simultaneously, with the output arbitration block splitting out messages by type or by source. Data going through the Aurora out port or the Trace port goes through

an additional translation block to put the data in the correct format. The actual Aurora formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The Buddy Device (BD) NAR supports the following client interface:

- Aurora type interface with clock domain transfer at the same frequency

95.2 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port and/or the Trace port. The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (1149.1) port.

95.3 Features

The NAR supports the following features:

- Connects 1 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Nexus out port for direct connection to pins (low-speed applications) or to another Nexus port controller
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Both output ports may operate simultaneously. Filtering logic sorts messages to determine destination port.
- Partitionable main queue makes it possible to simultaneously service multiple output ports. It can also be used to devote more bandwidth to high-priority message-types or split trace stream between on-chip memory and trace port.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client
- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message
- Programmable Global stall signal on debug tool indication of memory overflow
- Event generation on External Trace Memory (overlay) programmable full
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

95.4 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 2057. NAR registers

Register name	Read/ Write	JTAG index (NAR relative)	Location
NAR_CR—NAR Control Register	R/W	0	on page 3797
NAR_ST—NAR Status Register	R	1	on page 3799
NAR_SFR—Source Filter Register	R/W	2	on page 3801
NAR_TFR—Type Filter Register	R/W	3	on page 3802
NAR_TBALO—Trace memory port base address, low	R/W	4	on page 3803
NAR_TBAHI—Trace memory port base address, high	R/W	5	on page 3803
NAR_TCR—Trace Memory bus control register	R/W	6	on page 3805
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)	on page 3806
NAR_CSSR—Client Suppress Status Register	R	B	on page 3807
NAR_CDR—Client Disable Register	R/W	C	on page 3808
NAR_NAPCR—Nexus Aurora Phy Control Register	R/W	D	on page 3810
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E	on page 3810

95.4.1 Register descriptions

95.4.1.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following figure shows the format of the NAR_CR.

Figure 2204. NAR control register (NAR_CR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG	SED	SRD	TBW	AQP	0	0	0	0	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	MSP		0	ALT		0	AFA	0	FSD	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CR is described in the following table.

Table 2058. NAR_CR field descriptions

Field	Description
31 NEN	NAR Enable. 0 NAR is disabled unless HBDP is independently enabled. See Section 95.5.2.2, Disabled for more information. 1 NAR is enabled.
30 NCD	NAR Communication Disable. 0 NAR monitors clients for activity 1 NAR communication with clients is disabled
29 TEN	Timestamp Enable. 0 Timestamp packet is not included in messages 1 Timestamp packet is included in messages from all client including NAR generated messages
28 TRS	Timestamp Reset. 0 to 1 transition: reset timestamp counter in NAR
27–26 TSG	Timestamp Granularity (Mode). <i>Note: Some clients may only support timestamps on every message.</i> 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 32nd message
25 SED	Stall Error Disable. 0 NAR injects dummy complete message when active client stalls 1 NAR does not inject dummy complete message

Table 2058. NAR_CR field descriptions (continued)

Field	Description
24 SRD	Stall Recovery Disable. 0 Stalled receive queue resumes operation when its client comes back on line 1 Stall error recovery is disabled
23 TBW	Tracebuffer Wrap. 0 NAR does not capture new data when main queue is full 1 NAR begins overwriting main queue from the beginning upon queue overflow
22–21 AQP	Alternate Queue Partition. 00 Queue partitioning is disabled 01 Reserved 10 1/2 of queue is dedicated to alternate partition 11 Reserved
20	Reserved and must be written with 0
19–16	Reserved and must be written with 0s
15–13 WPE	Watchpoint Enable 000 NAR watchpoint messaging is disabled xx1 NAR generates a watchpoint message upon a sync event x1x NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event 1xx NAR generates a watchpoint message upon a suppress-mode event
12	Reserved
11	Reserved. This bit must be written with 0.
10–9 MSP	Message Select Port. Defines which port is the target for message type/source filtering. Only active in two-port applications. 00 No message select port. 0–1 port is active. 01 HBDP is the MSP 10 Trace memory bus is the MSP 11 Nexus is the MSP
8	Reserved
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications. 00 No alternate port. Running in trace-buffer mode. 01 HBDP is the ALT 10 Trace memory bus is the ALT 11 Nexus is the ALT
5	Reserved
4 AFA	Automatic Flush Accumulators. 0 Output queue accumulators do not flush stranded beats 1 Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.
2 FSD	Frame Serial Data. 0 Framing disabled. Data is streamed through the Aurora out port. 1 Framing enabled. Start/End frame symbol inserted between every 256 Aurora words.

Table 2058. NAR_CR field descriptions (continued)

Field	Description
1 PUS	Power-Up Suppressed. 0 NAR powers up active 1 NAR powers up in global suppressed mode
0 NSR	NAR Soft Reset. 1 Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

95.4.1.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following figure shows the format of the NAR_ST.

Figure 2205. NAR status register (NAR_ST)

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				MPF	MLV		APF	ALV		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

The NAR_ST register is described in the following table.

Table 2059. NAR_ST field descriptions

Field	Description
31 NEN	NAR Enabled. 0 NAR is disabled and inactive 1 NAR is enabled with clocks on
30–28 NUM	NAR Mode. 000 NAR is in disabled mode 001 NAR is in global suppressed mode 010 NAR is in communication-disabled mode 011 NAR is in tracebuffer mode 100 NAR is in active transmit mode 101 Reserved 110 Reserved 111 NAR is in run-control mode

Table 2059. NAR_ST field descriptions (continued)

Field	Description
27–24 NCE	NAR Config Error. See Table 2074 for coding
23 MPF	MSP Partition Full. 0 MSP partition not full 1 MSP partition full
22–21 MLV	MSP Partition Queue Level. This field is available only when both output ports are enabled i.e. the main queue is partitioned. 00 0% < queue level < 25% 01 25% < queue level < 50% 10 50% < queue level < 75% 11 75% < queue level < 100%
20 APF	ALT Partition Full. 0 ALT partition not full 1 ALT partition full
19–18 ALV	ALT Partition Queue Level. 00 0% < queue level < 25% 01 25% < queue level < 50% 10 50% < queue level < 75% 11 75% < queue level < 100%
17–16	Reserved
15 HPR	HBDP Ready. 0 HBDP (Aurora Out port) is not ready 1 HBDP is ready and accepts data for transmission
14 HPE	HBDP Error. 0 No HBDP error 1 Error condition exists on HBDP
13 MPE	Trace memory bus Error. 0 No Trace memory bus error 1 Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow. 0 No Trace memory bus overflow 1 Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved
10 TOV	Tracebuffer Overflow. 0 No tracebuffer overflow 1 The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer of ALT partition (last valid write + 1)

95.4.2 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters determine which port receives a message, based on the message

source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results. See [Section 95.5.6, Output arbitration](#), for details on how the NAR arbitrates between multiple output ports. These registers should only be set in the BD NAR because partitioning is not supported in the PD NAR.

95.4.2.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Figure 2206. Source filter register (NAR_SFR)

Register index: 2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFI	FE0	SF0				0	0	0	FE1	SF1				0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	SF2				0	0	0	FE3	SF3				0	ACC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SF_n), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2060. NAR_SFR field descriptions

Field	Description
31 SFI	Source Filter Invert. 0 Match if message source equals any of the enabled filters 1 Match if message source equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. SF0 ignored 1 Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field. See Section 95.5.6, Output arbitration for an example.
25–23	Reserved.
22 FE1	Enable Filter 1. 0 Filter 1 disabled. SF1 ignored 1 Filter 1 enabled and contained in SF1

Table 2060. NAR_SFR field descriptions (continued)

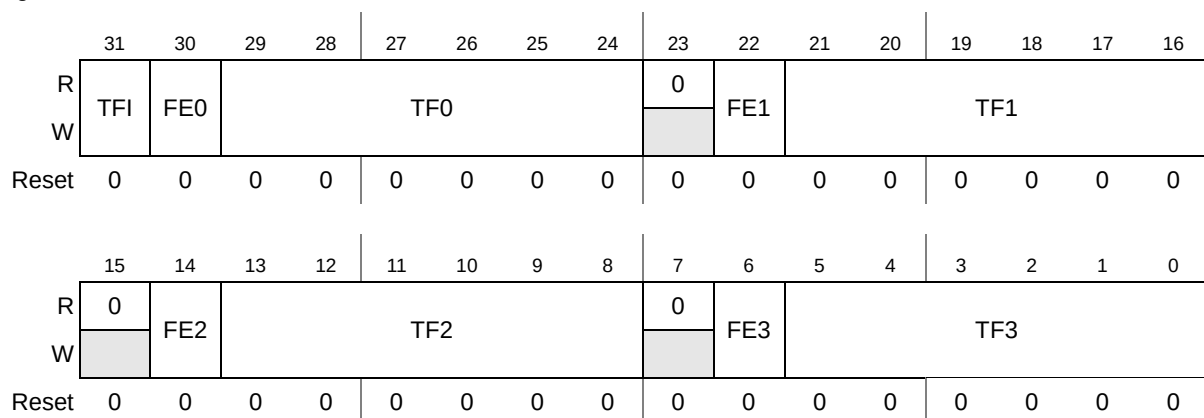
Field	Description
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field. See Section 95.5.6, Output arbitration for an example.
17–15	Reserved
14 FE2	Enable Filter 2. 0 Filter 2 disabled. SF2 ignored 1 Filter 2 enabled and contained in SF2
13–10 SF2	Source Filter 2. The message source trace ID is compared with this field. See Section 95.5.6, Output arbitration for an example.
9–7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. SF3 ignored 1 Filter 3 enabled and contained in SF3
5–2 SF3	Source Filter 3. The message source trace ID is compared with this field. See Section 95.5.6, Output arbitration for an example.
1	Reserved
0 ACC	OR source and type filters. 0 Match if source AND type filters match 1 Match if source OR type filters match

95.4.2.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Figure 2207. Message type filter register (NAR_TFR)

Register index: 3



The NAR_TFR is described in the following table.

Table 2061. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field. See Section 95.5.6, Output arbitration for an example.
23	Reserved
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field. See Section 95.5.6, Output arbitration for an example.
15	Reserved
16 FE2	Enable Filter 2. 0 Filter 2 disabled. TF2 ignored 1 Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field. See Section 95.5.6, Output arbitration for an example.
7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. TF3 ignored 1 Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field. See Section 95.5.6, Output arbitration for an example.

95.4.3 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

95.4.3.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can be transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following figures show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Figure 2208. Trace memory base address register NAR_TBAHI

Register index: 5

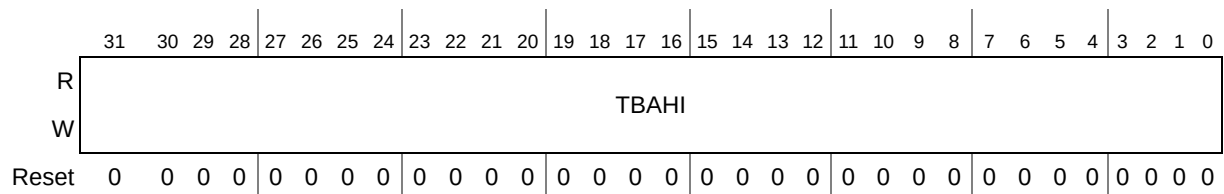
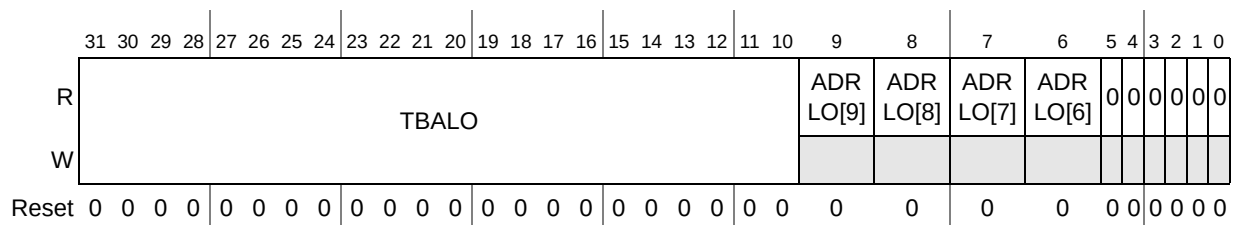


Figure 2209. Trace memory base address register NAR_TBALO

Register index: 4



The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 2062. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 2063. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)

Table 2063. NAR_TBALO field descriptions (continued)

Field	Description
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

95.4.3.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following figure shows the format of the NAR_TCR.

Figure 2210. Trace memory bus control register (NAR_TCR)

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	BTM	BTS		BTH		0	0	0	TBW	MXFR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	MXFR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TCR is described in the following table.

Table 2064. NAR_TCR field description

Field	Description
31 BTM	Block Transfer Mode. 0 Data for the Trace Memory bus port is transmitted one word at a time, upon first availability 1 Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29 BTS	Block Transfer Size. 00 Block transfer size = 64 bytes 01 Block transfer size = 128 bytes 10 Block transfer size = 256 bytes 11 Block transfer size = 384 bytes

Table 2064. NAR_TCR field description (continued)

Field	Description
28–27 BTH	Block Transfer Threshold. 00 Full threshold. Number of entries in partition must equal block transfer size before block transfer starts 01 Begin transfer when block is 25% full 10 Begin transfer when block is 50% full 11 Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.
32 TBW	Trace Memory Bus Block Wrap. 0 NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached 1 NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0 MXFR	Trace Memory Bus Max Transfer Size. Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.

95.4.4 Suppress mode triggers register (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in [Section 95.5.4, Suppress mode](#). The suppress mode triggers are only available on the PD NAR and do not apply to the BD NAR. The registers are present in the BD NAR, but input suppress triggers are not connected. Each trigger can be programmed to suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each Suppress Trigger Control Register, shown in the following figure, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Figure 2211. Suppress trigger control registers (NAR_STCR1–4)

Register index: 7–A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LST															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2065. NAR_STCR field descriptions

Field	Description
31 GST	Global Suppress Trigger. 0 The associated trigger cannot cause NAR to enter global suppress mode 1 The associated trigger is enabled for global suppress mode
29–16	Reserved
15–0 LST	Local Suppress Trigger. 0000000000000000The associated trigger is disabled for local suppress mode for all clients xxxxxxxxxxxxxx1 Trigger is enabled for Client 1 local suppress xxxxxxxxxxxxxx1x Trigger is enabled for Client 2 local suppress xxxxxxxxxxxxxx1xx Trigger is enabled for Client 3 local suppress xxxxxxxxxxxxxx1xxx Trigger is enabled for Client 4 local suppress xxxxxxxxxx1xxxx Trigger is enabled for Client 5 local suppress xxxxxxxxxx1xxxxx Trigger is enabled for Client 6 local suppress xxxxxxxxxx1xxxxxx Trigger is enabled for Client 7 local suppress xxxxxxxxxx1xxxxxxx Trigger is enabled for Client 8 local suppress xxxxxxx1xxxxxxx Trigger is enabled for Client 9 local suppress xxxxxx1xxxxxxx Trigger is enabled for Client 10 local suppress xxxxx1xxxxxxx Trigger is enabled for Client 11 local suppress xxxx1xxxxxxx Trigger is enabled for Client 12 local suppress xxx1xxxxxxx Trigger is enabled for Client 13 local suppress xx1xxxxxxx Trigger is enabled for Client 14 local suppress x1xxxxxxx Trigger is enabled for Client 15 local suppress 1xxxxxxx Trigger is enabled for Client 16 local suppress All other options Reserved

95.4.4.1 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status (NAR_CSSR), shown in the following figure, can then be read to determine which clients are suppressed, and which are active.

Figure 2212. Client suppress status register (NAR_CSSR)

Register index: B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	LSS																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2066. NAR_CCSR field descriptions

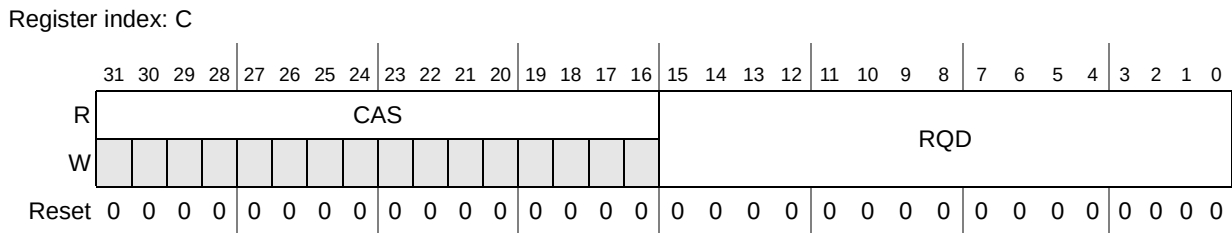
Field	Description
31–16 LSS	Local Suppress Status. 0000000000000000No client is in local suppress mode xxxxxxxxxxxxxxxx1 Client 1 is suppressed xxxxxxxxxxxxxxxx1x Client 2 is suppressed xxxxxxxxxxxxxxxx1xx Client 3 is suppressed xxxxxxxxxxxxxxxx1xxx Client 4 is suppressed xxxxxxxxxxxxx1xxxx Client 5 is suppressed xxxxxxxxxxxxx1xxxxx Client 6 is suppressed xxxxxxxxxx1xxxxxx Client 7 is suppressed xxxxxxxxx1xxxxxxx Client 8 is suppressed xxxxxxx1xxxxxxxx Client 9 is suppressed xxxxxx1xxxxxxxxx Client 10 is suppressed xxxxx1xxxxxxxxxx Client 11 is suppressed xxxx1xxxxxxxxxxx Client 12 is suppressed xxx1xxxxxxxxxxxx Client 13 is suppressed xx1xxxxxxxxxxxxx Client 14 is suppressed x1xxxxxxxxxxxxxx Client 15 is suppressed 1xxxxxxxxxxxxxxx Client 16 is suppressed All other options Reserved
15–0	Reserved

95.4.5 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following figure, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.

Note: Even though the BD NAR is described as only having one client, there is actually a second dummy client, client 2. To re-allocate the dummy RX queue to client 1 (the PD NAR), set the BD NAR client 2 RQD bit.

Figure 2213. Client disable register (NAR_CDR)



The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.



Table 2067. NAR_CDR field descriptions

Field	Description
31–16 CAS	Client Activity Status. xxxxxxxxxxxxxxxx1 Client 1 is active with new data xxxxxxxxxxxxxxxx1x Client 2 is active with new data xxxxxxxxxxxxxxxx1xx Client 3 is active with new data xxxxxxxxxxxxxxxx1xxx Client 4 is active with new data xxxxxxxxxxxxxxxx1xxxx Client 5 is active with new data xxxxxxxxxxxxxxxx1xxxxx Client 6 is active with new data xxxxxxxxxxxxxxxx1xxxxxx Client 7 is active with new data xxxxxxxxxxxxxxxx1xxxxxxx Client 8 is active with new data xxxxxxxxxxxxxxxx1xxxxxxx Client 9 is active with new data xxxxxxxxxxxxxxxx1xxxxxxx Client 10 is active with new data xxxxxxxxxxxxxxxx1xxxxxxx Client 11 is active with new data xxxx1xxxxxxxxxxxxx Client 12 is active with new data xxx1xxxxxxxxxxxxxx Client 13 is active with new data xx1xxxxxxxxxxxxxxx Client 14 is active with new data x1xxxxxxxxxxxxxxx Client 15 is active with new data 1xxxxxxxxxxxxxxx Client 16 is active with new data All other options Reserved
15–0 RQD	Receive Queue Disable. xxxxxxxxxxxxxxxx1 Disable receive queue for Client 1 xxxxxxxxxxxxxxxx1x Disable receive queue for Client 2 xxxxxxxxxxxxxxxx1xx Disable receive queue for Client 3 xxxxxxxxxxxxxxxx1xxx Disable receive queue for Client 4 xxxxxxxxxxxxxxxx1xxxx Disable receive queue for Client 5 xxxxxxxxxxxxxxxx1xxxxx Disable receive queue for Client 6 xxxxxxxxxxxxxxxx1xxxxxx Disable receive queue for Client 7 xxxxxxxxxxxxxxxx1xxxxxxx Disable receive queue for Client 8 xxxxxxxxxxxxxxxx1xxxxxxx Disable receive queue for Client 9 xxxxxxxxxxxxxxxx1xxxxxxx Disable receive queue for Client 10 xxxxxxxxxxxxxxxx1xxxxxxx Disable receive queue for Client 11 xxxx1xxxxxxxxxxxxx Disable receive queue for Client 12 xxx1xxxxxxxxxxxxxx Disable receive queue for Client 13 xx1xxxxxxxxxxxxxxx Disable receive queue for Client 14 x1xxxxxxxxxxxxxxx Disable receive queue for Client 15 1xxxxxxxxxxxxxxx Disable receive queue for Client 16 All other options Reserved

95.4.6 Nexus Aurora Phy control register (NAR_NAPCR)

The Nexus Aurora Phy control register provides configuration control to number of HBDP lanes of Nexus Aurora Phy and LVDS pad control signals. The following figure shows the format of the NAR_NAPCR.

Figure 2214. Nexus Aurora Phy control register (NAR_NAPCR)

Register index: D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	NUM_LANE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0		0	LVDS_TXOPT				LVDS_RX_TERM_EN		LVDS_BIAS_PRG		LVDS_RXOPT			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

The NAR_NAPCR is described in the following table.

Table 2068. NAR_NAPCR field descriptions

Field	Description
19–16 NUM_LANE	TX Lane Configuration. Number of enabled TX lanes. This field is device specific. For a description of this field see the Calibration and Debug configuration chapter which describes how the modules are configured.
11–8 LVDS_TXOPT	This is an internal characterization field.
7 LVDS_RX_TERM_EN	This is an internal characterization field.
6 LVDS_BIAS_PRG	This is an internal characterization field.
5–0 LVDS_RXOPT	This is an internal characterization field.

95.4.7 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the

partition related configuration for main queue (internal central queue). The following figure shows the format of the NAR_AHFPAR.

Figure 2215. Trace memory fill level and partition configuration register (NAR_AHFPAR)

Register index: E

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	GSE	0	0	0	GLOBAL_STALL_COUNT				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0				

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	AHB65kEN	AHB01kEN	PART_CFG		AHB_65K_Marker				AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_AHFPAR is described in the following table.

Table 2069. NAR_AHFPAR field description

Field	Description
31-25	Reserved and must be written with 0s
24 GSE	Global Stall Enable. Setting this bit enables the NAR to act on dci_evti[0] event to generate a global stall
23-21	Reserved and must be written with 0s
20-16 GLOBAL_STALL_COUNT	Global Stall Count. Value at this field determines how many cycles, in the NAR's operating clock, the global stall remains asserted on a dci_evti event. This is a five bit field so the maximum value is 31 cycles. See Section 95.5.13
15-12	Reserved and must be written with 0s
11 AHB65kEN	Enable 65K Marker Event
10 AHB01kEN	Enable 1K Marker Event
9-8 PART_CFG	Partition Configuration. 00 No partition 01 Reserved 10 50% partition (in two partition two configuration case) 11 Reserved

Table 2069. NAR_AHFPAR field description (continued)

Field	Description
7–4 AHB_65K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 65K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Section 95.5.14 <i>Note: This is used in BD as it has memory ranges from 65K to 983K</i>
3–0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Section 95.5.14 <i>Note: This is used in PD as it has memory ranges from 1K to 15K</i>

95.5 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

95.5.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state. This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)
3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/filtering when two output ports are enabled, enabling timestamp and so on.
4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note: *The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.*

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see [Section 95.5.4, Suppress mode](#). Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

95.5.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in this table.

Table 2070. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

95.5.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

95.5.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

95.5.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

95.5.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity

from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

95.5.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in this table on the incoming messages before transmitting to one of the output ports.

Table 2071. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8×32 -bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8×32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

95.5.3 Data reception and arbitration

The BD NAR supports 1 client (although a second dummy client is used to increase the RX buffer size for client 1). Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat). The number of beats varies from client to client. The number of beats (B_n) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There is one type of client supported at NAR client interface. One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits.

When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal

and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire $B_n \times 32$ data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment. Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

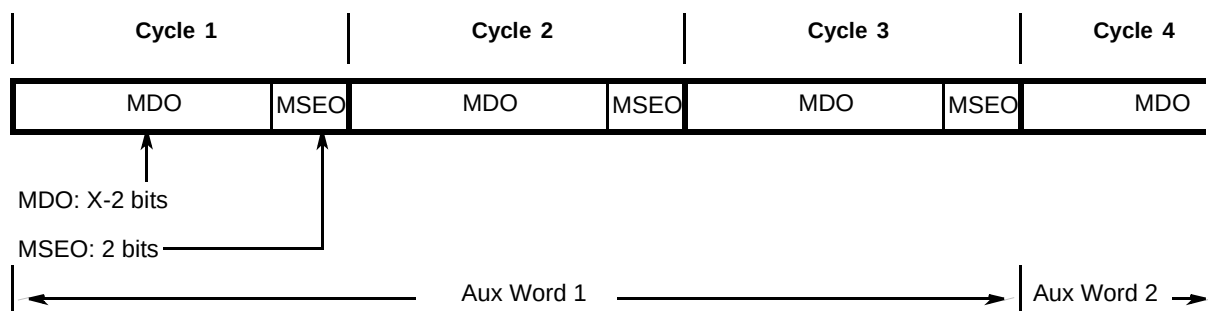


Figure 2216. AUX bus parsing

Each AUX word consists of B_n cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new

message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

95.5.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. Suppress mode is only supported in the PD NAR, it is not supported in the BD NAR. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to one

set of clients, while suppress trigger two can be assigned to another set of clients, and different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

95.5.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue. The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

95.5.6 Output arbitration

The NAR allows two output ports to be active at the same time. This feature is only available on the BD NAR. Typically, this feature is only useful if one of the ports is used to service messages of a specific type and/or messages from a particular client, while the other port supports all the rest of the message traffic. For example, the NAR could be set up to send in-circuit trace (ICT) traffic out the HBDP and to reserve the trace memory bus port for all other Nexus messages.

If two output ports are enabled, one of them must be designated as the Message Select Port (MSP). The MSP is the destination for those messages that satisfy a set of user-defined requirements as to the source and type of message. All other traffic is sent to the other enabled port. Even though the NAR supports three outputs (HBDP, Nexus, and trace

memory bus) only two can be used at the same time. Ports are enabled by setting the port enable bits in the BD NAR_CR. The MSP is designated by setting bits [10–9] and ALT is designated by setting bits [7–6]. Once the MSP has been specified, other bits in the NAR_CR, along with the Source Filter Register (NAR_SFR) and Type Filter Register (NAR_TFR), are used to determine which messages go to the MSP and which do not. These facilities can be used to build a logical expression that can filter messages based on up to four types and/or four sources.

The NAR_SFR can be programmed with up to four source codes, while the NAR_TFR can hold up to four message-type codes. Each filter also includes an enable bit to turn it on or off. A source match occurs if a messages source code matches one of the source filters or (if the SFI bit is set) if it does not match any of the source filters. Similarly, a type match occurs if a messages type code matches one of the type filters or (if the TFI bit is set) if it does not match any of the type filters. Finally, the ACC bit of the NAR_SFR determines when both source and type filters are used, whether the two are ANDed together (ACC = 0) or ORed together (ACC = 1). For example, to send ICT messages out the HBDP and all other traffic to the trace memory bus port:

- Set ALT = 10 in the NAR_CR—sets trace memory bus port as the alternate port
- Set MSP = 01 in the NAR_CR—sets HBDP as the message select port
- Set FE0 = 1 & TF0 = 100010 (0x22) in the NAR_TFR—enables type filter 0 to match ICT messages

To send everything but data trace from source 13 to the trace memory bus port, while all other traffic goes to the HBDP:

- Set ALT = 01 in the NAR_CR—sets HBDP as the alternate port
- Set MSP = 10 in the NAR_CR—sets trace memory bus port as the message select port
- Set TFI = 1—inverts type-filter logic
- Set FE0 = 1 & TF0 = 000101 (0x05) in the NAR_TFR to select data trace of size 8 bits
- Set FE1 = 1 & TF1 = 000110 (0x06) in the NAR_TFR to select data trace of size 16 bits
- Set FE2 = 1 & TF2 = 001101 (0x0d) in the NAR_TFR to select data trace of size 32 bits
- Set FE3 = 1 & TF3 = 001110 (0x0e) in the NAR_TFR to select data trace size of 64 bits
- Set FE0 = 1 & SF0 = 001101 (0x0d) in the NAR_SFR to select source trace ID = 13
- Set ACC = 0 in NAR_SFR so the final filter is $\sim(\text{data_trace}) \text{ AND } (\text{source} = 13)\text{HBDP}$

Data is written into the output queue at a maximum rate of one entry per cycle. If the current messages source and type pass the filters, then the message is marked for the MSP. When the entry is later pulled from the queue, it is then sent to the MSP. Unmarked entries are sent to the alternate port. Message destination must be recalculated on every end-of-message boundary, using the fixed-width SRC and TYPE fields at the beginning of each Nexus message.

95.5.7 Queue partitioning

It is possible to partition the queue into two sections, one that supports the MSP, and the other that supports the alternate port as shown in [Figure 2217](#). Messages are routed, based on the MSP bit field, to either the MSP queue partition or the alternate queue partition. This allows both ports to be active simultaneously, since messages can be taken directly from both partitions. Each partition uses an accumulator to store a full central queue width beats of data before writing into memory, since on any given cycle the full Nexus word out of receiver arbitration may be split between the two partitions. Queue partitioning is enabled by setting the AQP bits of the NAR_CR.

The main advantage of using a partitioned queue is that it can improve data throughput and performance when multiple ports are active. It allows the trace memory bus port to utilize block transfer mode, which also improves performance. The only drawback is that queue utilization efficiency can be negatively impacted.

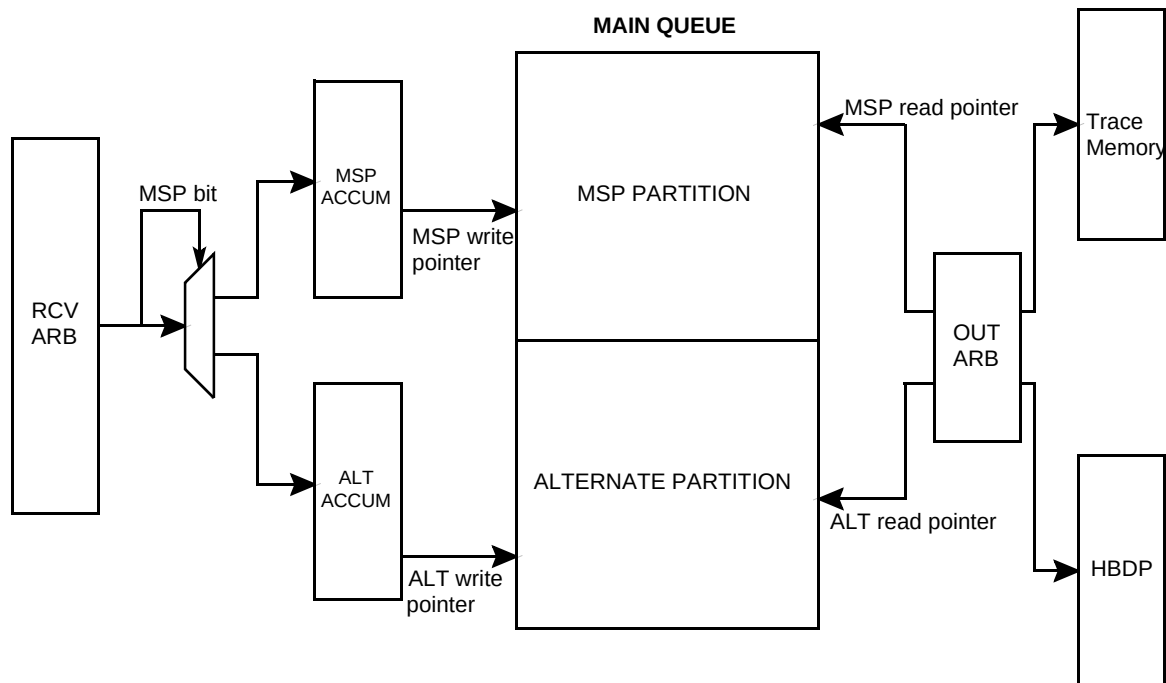


Figure 2217. Queue partitioning

95.5.8 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

95.5.8.1 Aurora format

Nexus messages are pre-translated before sending to the HBDP using the Aurora protocol. The NAR output is a stream of 16×4 -bit words (where 4 is the width of the NAR Aurora out port) that are further encoded by an external Aurora block before being sent for transmission on the Aurora (HBDP) port.

Entries from the output queue are chopped into chunks of 4×16 -bit words. The amount of data that can be transmitted per Aurora cycle depends on the number of lanes available in the HBDP (transmitted HBDP word = $16 \times$ number of lanes).

95.5.8.2 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and resizing in the likely case where the trace memory bus width does not equal the queue entry size.

95.5.8.3 Nexus format

If the Nexus port is used, entries are sent directly out of the queue with no translation or resizing. The Nexus output, Nexus input, and queue are all the same width.

The Nexus port also includes an EVTO output. This allows EVTO signalling functionality to be maintained in systems with multiple NARs. EVTO can also be used as a trigger by an external event processing block, to allow events such as client halt to occur based on NAR events such as tracebuffer overflow. Currently, the only event that can cause an EVTO assertion is an HBDP hard error/reset, since data is almost certainly lost when the HBDP link goes down, the NAR needs to communicate that back to the clients so that they can generate sync messages.

Many Nexus message types use address/data compression, wherein the compressed data is generated by XORing the current data with the previous data sent. This type of compression can only work if there is no data lost; if data is lost, the client must start over by sending a new sync message which contains the full uncompressed address or data.

95.5.9 Aurora interface

The external Nexus Aurora Link (NAL) contains all of the link-layer, PCS, and control logic for the HBDP channel. The Aurora data is transmitted from the NAR to the NAL for sending to the HBDP. Typically, the NAL powers up coming out of reset, but it can also be enabled by the NAR by turning on the NAR with the HBDP enabled. The reverse is true as well. If the NAL is active while the NAR is disabled, then the NAR powers up to communication-disabled mode.

Data flow between the NAR and the NAL uses a simple, synchronous interface. If the NAL is active, it indicates this by a ready signal from the NAL. If Aurora is configured as output it is indicated by a signal from the NAR.

The Aurora interface can either be operated in framing or streaming mode. Streaming mode provides the highest performance and lowest communication overhead through the link. Whereas framing mode provides greater robustness and improved error-handling at the receive side of the link. When framing mode is enabled, the NAR encapsulates the Aurora data in frames consisting of 256 Aurora chunks; that is, after every 256 Aurora transmissions, the NAR asserts end of frame and start of frame to mark the end of the last frame and the beginning of the next frame. In either mode, the NAR always asserts a start-of-frame when beginning transmission, and an end-of-frame when its output queue empties, to notify the channel partner of the end of valid data.

95.5.10 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not

available. The trace memory bus access are generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (i.e. unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)
- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be $0x100 \times 0x40 = 0x4000$ bytes. The dedicated debug address range would then be 0x0D00_0000 – 0x0D00_3FFF.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

95.5.11 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- **Device_ID**—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.
- **Watchpoint**—The NAR generates its own watchpoint match message under the following circumstances as shown in [Table 2075](#).
- **NAR_Error (TCODE = 8)**—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
 - HBDP error
 - Illegal configuration
 - Stall Error
 - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 2072. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

Table 2072. NAR message formats (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Error Message ⁽¹⁾	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC ⁽²⁾	NAR source ID
	4	4	fixed	ETYPE	See Table 2073
	12	12	fixed	ECODE	See Table 2074
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message ⁽³⁾	6	6	fixed	TCODE	Value = 15
	6	6	fixed	SRC ⁽²⁾	NAR source ID
	6	6	fixed	WPHIT	See Table 2075
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2011 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2011 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

95.5.11.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in this table.

Table 2073. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see Table 2074 .

Table 2073. NAR error types (continued)

ETYPE	Description	Classification	NAR action	Comment
0011 (h'3)	Trace memory bus port read/write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes. ⁽¹⁾
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See Table 2074	The specific configuration error is conveyed in bits 0–3 of the ECODE field (Table 2074), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.
1111 (h'F)	Stall error	Minor	—	—

1. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in this table.

Table 2074. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.
00001xxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.

Table 2074. Configuration error codes (continued)

ECODE	ETYPE	Description	NAR action	Comment
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering and/or queue partitioning disabled	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

95.5.11.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

Table 2075. Watchpoint Hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

95.5.12 Timestamp function

The NAR and all of its clients are required to transmit timestamps, to allow precise ordering of events. All clients synchronous or asynchronous uses NAR timer value broadcasted over a bus, this ensures that all timestamps are in sync as shown in [Figure 2218](#). The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

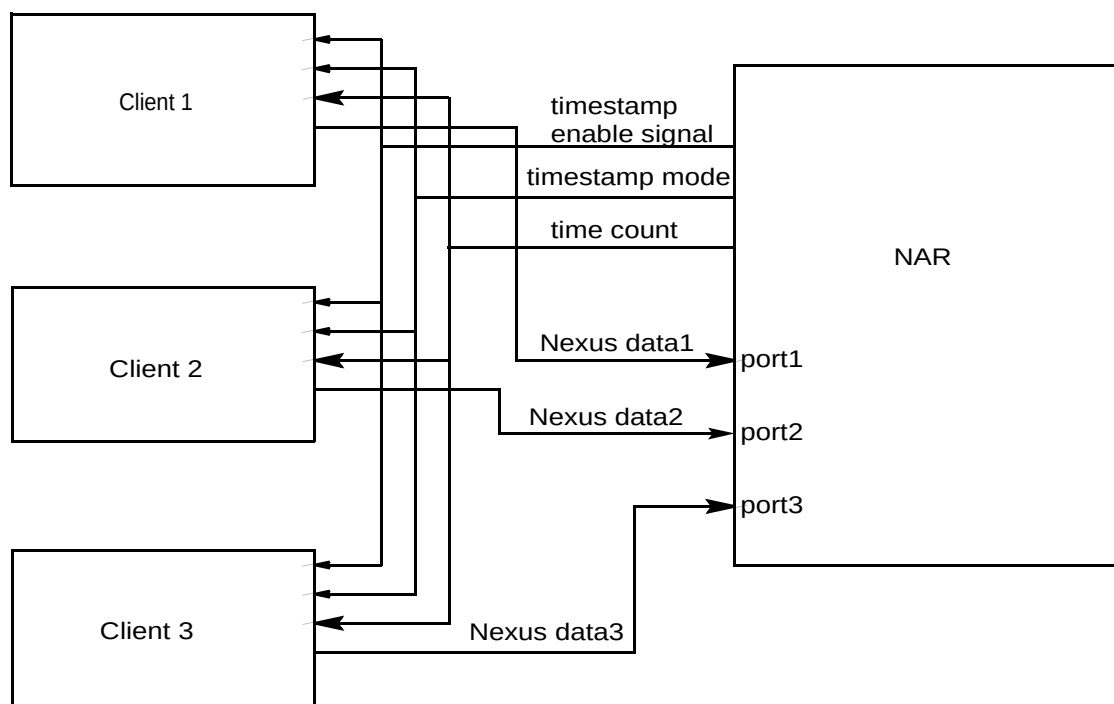


Figure 2218. Timestamp architecture

Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The Time stamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

95.5.13 Global (Development tool client) stall control

The external development tool has finite memory. To prevent data loss when this memory is full, the tool gives an event to the device. The DCI block processes this and forwards to

NAR block. The dci_evti[0] carries this information. NAR detects the falling edge of dci_evti[1] and loads a counter with a configurable value programmed in GLOBAL_STALL_COUNT field of NAR_AHFPAR. This counter starts decrementing immediately, reaches to zero and stops. During the running period of the counter the Global Stall signal remains asserted thus indicating to stall the message interface to the NAL by that many cycles. The GLOBAL_STALL_COUNT field is five bits so client can be stalled up to 31 cycles.

95.5.14 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:
 - AHB start (base) address—Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)
 - Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
 - Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)
- Thus, the total block size is $(0xC0 \times 0x40) = 0x3000$ bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
 - Number of 1K blocks in 0x3000 bytes = $(3000 \text{ right shift by } 10 \text{ bits}) = C = 12$
 - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- $\text{NAR_AHFPAR}[3:0] = (\text{NAR_TBALO}[13:10] + 0xB) = (0x7 + 0xB) \times 2$ (upper bits truncated due to width) and enable this with NAR_AHFPAR[10] = 1.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

Similarly for 65 KB boundary matching NAR_AHFPAR[7:4] needs to be programmed and the corresponding enable bit needs to be set.

95.5.15 Configuration/debug interface JTAG (1149.1)

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in [Table 2076](#). Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

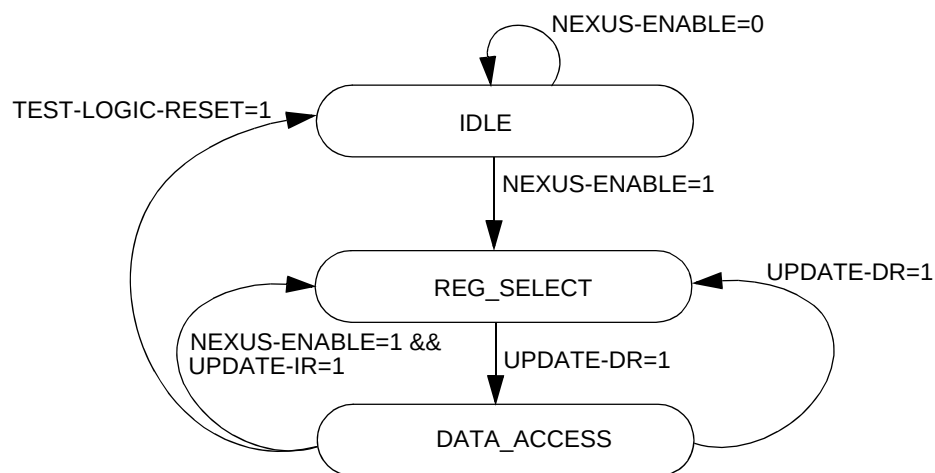


Figure 2219. Nexus controller state machine

Table 2076. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

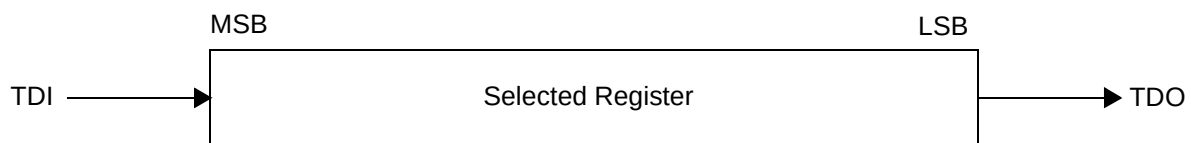


Figure 2220. Shifting data into register

95.5.15.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

95.5.15.2 Loading NEXUS-ENABLE instruction

Access to the NAR registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in [Figure 2219](#), transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 2077. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS-ENABLE instruction.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

95.5.15.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register

address index, as illustrated in [Figure 2221](#). The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in [Table 2057](#).

MSB	LSB
7-bit register index	R/W

Figure 2221. Nexus command format for register selection

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

This table illustrates a sequence which writes a 32-bit value to a register.

Table 2078. Writing a 32-bit value to a register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

95.5.16 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface
 - Early Write (for higher BW in special module)
 - Configurable 32/64-bit address bus width
 - 64-bit data bus width
 - Transfer Types
 - IDLE, NONSEQ, SEQ
 - Single Transfers
 - Burst Transfers
 - Configurable support for INCR
 - Configurable support for INCR4, INCR8, INCR16, WRAP4
 - Response Type
 - OKAY, ERROR
 - Transfer Size
 - Byte, Half Word, Word, Double-Word
 - Protection Control
 - Cacheable/ Non Cacheable Access Control
 - User/Privilege Access
- Endianess Support
 - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
 - Total of two transactions (One plus one ongoing)

95.5.16.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

95.5.16.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes have been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

95.5.16.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64-bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

95.5.17 Aurora client controller

This block converts the Aurora client protocol from the PD NAR into an internal client protocol that is accepted by the NPC block. The Nexus beat size must be same at both interfaces. The relation between the Aurora client data width and the NPC client interface data width must be maintained as described here.

The Aurora data width is defined as the Aurora width $\times$ 16 bits, where 16 bits comprise one Aurora word. The Nexus client data width is defined as $B_n \times 32$, where $32 = \text{MDO bits} + 2 \text{ MSEO bits}$ and B_n is the number of beats.

Thus: Aurora data width = the Nexus client data

$$\text{Aurora width} \times 16 = B_n \times 32$$

$$\text{Aurora width} = 2 \times B_n.$$

This block works in a similar way to the legacy client controller block except for the interface level control. The Aurora client has the information about the FIFO size. It sends as many data as the FIFO size then waits for the data accept signal from this block. The controller gives the data accept only when a read occurs and sends data to the NPC interface. The FIFO act as a storage and serves the purpose of clock domain transfer as well.

95.6 Initialization information

The NAR is initialized by asserting the block reset signal for a minimum of one cycle of the slowest clock. Reset clears all latches to 0, with the only exception being the output queue, whose initialization state depends on the reset behavior of the compiled memory used. Upon exiting reset, the NAR is in the disabled state, as described in [Section 95.5.2.1, Reset](#).

Pointers and state machines are initialized by the starting of NAR functional clocks (which occurs either upon the setting of the NAR enable bit or upon the assertion of the Phy enable signal) or by an NAR soft reset.

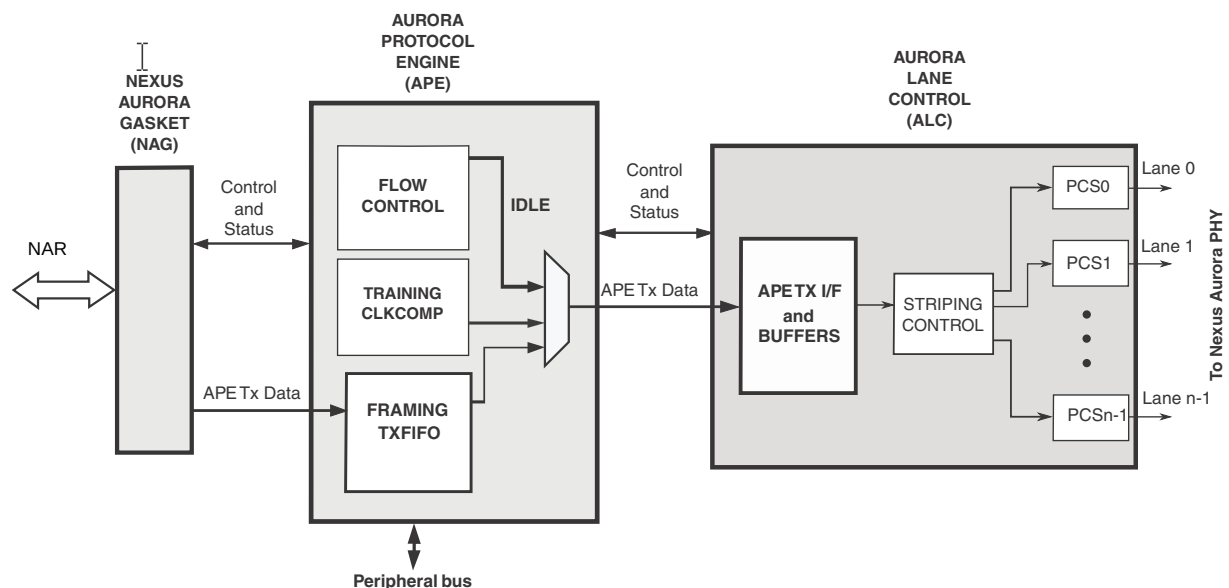
95.7 Application information

The base application for the NAR is collecting and collating Nexus trace data. Programming examples for how to set up the NAR for trace are given in [Section 95.5.6, *Output arbitration*](#).

96 Buddy Device—Nexus Aurora Link (NAL)

The Aurora Protocol is used to send debug information over a high-speed serial link out of the device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer SerDes and transceivers; it includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol. [Figure 2222](#) details the NAL block diagram. The number of lanes, n , is 4.

Figure 2222. NAL block diagram



Debug data in the Nexus format from multiple clients is collected by the Nexus Aurora Controller (NAR); messages are then packetized and sent across an asynchronous clock boundary-crossing gasket to the Aurora Protocol Engine, which performs all the logical layer functions and flow control; data from the protocol engine is sent on to the Aurora Lane Control (ALC) block, which performs data-striping and sizing, based on the number of lanes selected for debug; it also performs the PCS functions of the link. The PCS drives the Nexus Aurora Physical (NAP) interface. The NAP, in conjunction with the NAL, supports one-way simplex high-speed serial communication with an external debugger.

The NAL is composed of three main blocks: the NAR/Aurora Gasket, the Aurora Protocol Engine, and Aurora Lane Control. [Table 2079](#) lists NAL functionality by block and lists the high level features for the Nexus Aurora PHY.

Table 2079. NAL and NAP feature summary

Sub-component	Feature
NAR Aurora Gasket (NAG)	– Transmit FIFO to buffer data as it crosses the NAR / NAL clock-domain boundary and Transmit FIFO space available indication to NAR. The NAR uses Transmit FIFO space available indication(s) to determine if data can be sent to the NAL. – Platform to Domain clock domain crossing. The NAR is on the core clock domain while the NAP is on a domain sourced from the external PHY clock input signals.
Aurora Protocol Engine (APE)	– Link training – Per the Aurora Protocol specification, clock-compensation, and data framing – Cyclic Redundancy Check (CRC) generator
Aurora Link Control (ALC)	– Striping of data stream from the Aurora core across all available lanes in two-byte chunks – PCS functionality for NAL
Nexus Aurora PHY (NAP)	– Physical transmission of serial data to device pins – External clock source for SerDes clock domain

96.1 Transmit operation

During transmit operation, the NAR collects debug data from Nexus clients over a 32-bit interface: 30 bits are used for MDO and 2 bits are used for MSEO information. The NAR places MDO data into a payload (maximum payload size depends on the number of lanes, each lane has a maximum of 7424 symbols before terminating the payload) and adds the following:

- 16-bit start symbol
- 16-bit end symbol
- 16-bit CRC if enabled before end of previous symbol

Each packet is transferred from the NAR to the NAR/NAL gasket, through the Aurora Protocol Engine, the Aurora Lane Control and finally out to the device SerDes/LVDS pins through the NAP.

After each payload transfer, the NAL adds a clock compensation sequence needed for downstream SerDes devices to remain synchronized to the MCU.

Note: Symbol counting for clock compensation starts just prior to the verification symbols; therefore, training symbols are excluded when counting.

96.2 Nexus Aurora formatting

The Aurora physical interface is the transport mechanism for Nexus messages. The Nexus messages are formatted by the Aurora formatter and striped into Aurora lanes. The Nexus messages themselves encapsulated into a virtual Nexus port that consists of a 30-bit Message Data Output (MDO) port and a 2-bit Message Start/End Output (MSEO) signal. The MSEO portion of the port is the two least significant bits (LSB) of the 32-bit Aurora message. The rest of the 32-bit Aurora frame consists of 30 MDO bits that are added LSB first.

96.3 Static training

The NAL supports static training. The NAL performs Static training by using programmable timers to determine how long to stay in each stage of the training process. In static training, there is no communication between the channel partners to exchange training status; rather the NAL sends the align/bond/verify sequence to its partner for a set amount of time sufficient for the receive partner to complete the training stage.

Note: Prior to initialization, the NAL must be reset to its POR state; to do this, assert the block reset. See the RST bit field description in the NAL General Control Register (NAL_GCR).

The following steps enable static training:

1. Program the counters in the NAL_TCR and set the STE bit
2. Set the RST bit

This resets the channel and start the training routine. The time spent in each stage can range from 16 to 3072 cycles; once a counter times out, a counter time-out signal is asserted to the Aurora Protocol Engine (APE) to prompt it to move to the next stage. Additionally, it is possible to hold the training procedure in any of the stages indefinitely, by setting one of the three hold bits (AHD/BHD/VHD). When set, these bits cause the channel to sit in Align/Bond/Verify until unset, allowing a user to interactively step through the training procedure by monitoring the status at the debugger side and then stepping the device side to the next stage.

During the Aurora training sequence, the NAL generates a specific set of character sequences per the Aurora protocol specification which are to be transmitted simultaneously on all lanes.

96.4 Programmable registers

The Nexus Aurora Link (NAL) module provides a Nexus parallel to Aurora formatting for the MCU trace information. There are several registers to control both transmit and receive functionality. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 2080. NAL memory map

Offset	Register name	Width (in bits)	Access	Reset value
0h	NAL General Status Register (NAL_GSR)	32	R	0000_8000h
8h	NAL General Control Register (NAL_GCR)	32	R/W	0000_0000h
Ch	NAL Training Control Register (NAL_TCR)	32	R/W	0000_0000h

96.4.1 Nexus Aurora Link General Status Register (NAL_GSR)

Figure 2223. Nexus Aurora Link General Status Register (NAL_GSR)

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			TXCFG			0									AS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2081. NAL_GSR field descriptions

Field	Description
31–13	Reserved
12–10 TXCFG	TX Lane Configuration Number of enabled TX lanes. These bits are read-only and are controlled external to the NAL. 000 Reserved 001 2 TX lanes 010 Reserved 011 4 TX lanes 100 Reserved 101 Reserved 110 Reserved 111 Reserved
9–1	Reserved
0 AS	Aurora Status 0 Aurora is not enabled. 1 Aurora is enabled.

96.4.2 NAL General Control Register (NAL_GCR)

The NAL_GCR configures the behavior of the APE. It can be programmed to reset the Aurora channel, or to configure simplex/duplex initialization and to enable debug features.

Figure 2224. NAL General Control Register (NAL_GCR)

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			0													
W	RST	ALCPD													FCM	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		0	SLD			0	BOOE	BOOD				CRCEB	CCOEN	0	
W		PCRST														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2082. NAL_GCR field descriptions

Field	Description
31 RST	Reset the Aurora Channel 0 to 1 transition resets the channel and begins the training sequence. 0 normal operation 1 transition from 0 to 1 resets channel and starts training sequence
30 ALCPD	ALC/NAL Power Down Gate off clocks to the NAL blocks (ALC and APE). 0 NAL is powered down and inactive. 1 NAL is enabled with all clocks running provided other mechanisms of powerdown (such as alc_num_lanes) are not in effect.
29–18	Reserved This read-only bitfield is reserved and always has the value zero.
17 FCM	Flow Control Mode 0 Completion Mode 1 Immediate Mode
16–15	Reserved and must be written with zero.
14 PCRST	Protocol Converter Reset Resets the ALC when asserted.
13	Reserved This read-only bit is reserved and always has the value zero.
12–10 SLD	Symbol Lock Delay 000 Allow 3 characters to accumulate in elastic buffer before starting to read 001 Allow 4 characters to accumulate in elastic buffer before starting to read 010 Allow 5 characters to accumulate in elastic buffer before starting to read 011 Allow 6 characters to accumulate in elastic buffer before starting to read 100 Allow 7 characters to accumulate in elastic buffer before starting to read 101 Allow 8 characters to accumulate in elastic buffer before starting to read 110 Allow 9 characters to accumulate in elastic buffer before starting to read 111 Allow 10 characters to accumulate in elastic buffer before starting to read

Table 2082. NAL_GCR field descriptions (continued)

Field	Description
9	Reserved This read-only bit is reserved and always has the value zero.
8 BOOE	Bond Offset Override Enable 0 ALC channel bond attempts to determine channel bond offset. 1 ALC uses user override setting for channel bond offset.
7–4 BOOD	Bond offset override data Only valid when BOOE is set. 0000/1000 use 0-cycle offset 0001 use -7 cycle offset 0010 use -6 cycle offset 0011 use -5 cycle offset 0100 use -4 cycle offset 0101 use -3 cycle offset 0110 use -2 cycle offset 0111 use -1 cycle offset 1001 use +1 cycle offset 1010 use +2 cycle offset 1011 use +3 cycle offset 1100 use +4 cycle offset 1101 use +5 cycle offset 1110 use +6 cycle offset 1111 use +7 cycle offset
3 CRCEB	CRC Enable 0 CRC generation and checking is disabled for TX and RX data. 1 CRC is inserted for TX frames and CRC is checked for RX paths.
2 CCOEN	Clock Compensation Override Enable This is a debug feature which allows the duration between clock compensation sequences to be reduced to 1280 cycles. 0 Clock compensation counter times out at 7424 cycles 1 Clock compensation counter times out at 1280 cycles
2	Reserved and must be written with zero.
1–0	Reserved This read-only bitfield is reserved and always has the value zero.

96.4.3 NAL Training Control Register (NAL_TCR)

The NAL_TCR supports static training as described in [Section 96.3, Static training](#).

Figure 2225. NAL Training Control Register (NAL_TCR)

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STE	AHD	BHD	VHD	Reserved					ATC				Reserved				BTC				Reserved					VTC					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2083. NAL_TCR field descriptions

Field	Description
31 STE	Static Training Enable 0 Reserved 1 Timer-based training method is used, ie. RX data is ignored and training is established based on ATC/BTC/VTC timers.
30 AHD	Hold in Align 0 Follow Aurora training sequence 1 Remain in align until this bit is cleared (applies even during duplex-mode training)
29 BHD	Hold in Bond 0 Follow Aurora training sequence 1 Remain in bond until this bit is cleared (applies even during duplex-mode training)
28 VHD	Hold in Verify 0 Follow Aurora training sequence 1 Remain in verify until this bit is cleared (applies even during duplex-mode training)
27–23	Reserved and must be written with zero.
22–19 ATC	Align Timer Count 0000 Remain in align for 16 cycles 0001 Remain in align for 24 cycles 0010 Remain in align for 32 cycles 0011 Remain in align for 48 cycles 0100 Remain in align for 64 cycles 0101 Remain in align for 96 cycles 0110 Remain in align for 128 cycles 0111 Remain in align for 192 cycles 1000 Remain in align for 256 cycles 1001 Remain in align for 384 cycles 1010 Remain in align for 512 cycles 1011 Remain in align for 768 cycles 1100 Remain in align for 1024 cycles 1101 Remain in align for 1536 cycles 1110 Remain in align for 2048 cycles 1111 Remain in align for 3072 cycles
18–14	Reserved

Table 2083. NAL_TCR field descriptions (continued)

Field	Description
13–10 BTC	Bond Timer Count 0000 Remain in bond for 16 cycles 0001 Remain in bond for 24 cycles 0010 Remain in bond for 32 cycles 0011 Remain in bond for 48 cycles 0100 Remain in bond for 64 cycles 0101 Remain in bond for 96 cycles 0110 Remain in bond for 128 cycles 0111 Remain in bond for 192 cycles 1000 Remain in bond for 256 cycles 1001 Remain in bond for 384 cycles 1010 Remain in bond for 512 cycles 1011 Remain in bond for 768 cycles 1100 Remain in bond for 1024 cycles 1101 Remain in bond for 1536 cycles 1110 Remain in bond for 2048 cycles 1111 Remain in bond for 3072 cycles
9–4	Reserved
3–0 VTC	Verify Timer Count 0000 Remain in verify for 16 cycles 0001 Remain in verify for 24 cycles 0010 Remain in verify for 32 cycles 0011 Remain in verify for 48 cycles 0100 Remain in verify for 64 cycles 0101 Remain in verify for 96 cycles 0110 Remain in verify for 128 cycles 0111 Remain in verify for 192 cycles 1000 Remain in verify for 256 cycles 1001 Remain in verify for 384 cycles 1010 Remain in verify for 512 cycles 1011 Remain in verify for 768 cycles 1100 Remain in verify for 1024 cycles 1101 Remain in verify for 1536 cycles 1110 Remain in verify for 2048 cycles 1111 Remain in verify for 3072 cycles

97 Buddy Device—Nexus Aurora PHY (NAP)

97.1 Introduction

The Nexus Aurora PHY (NAP), in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. Configuration settings in the Nexus Aurora Router (NAR) Nexus Aurora Phy control register (NAR_NAPCR) specify the NAP and NAL lane configuration. The NAL performs all logical layer functions, flow control, data-striping and sizing based on the number of data lanes available for use, and physical coding sublayer functions on the data being sent to the NAP for transmission. The NAP receives 8b/10b encoded data from the NAL then serializes and transmits this data through a given PHY lane's Low Voltage Differential Signaling (LVDS) buffers to an external debugger.

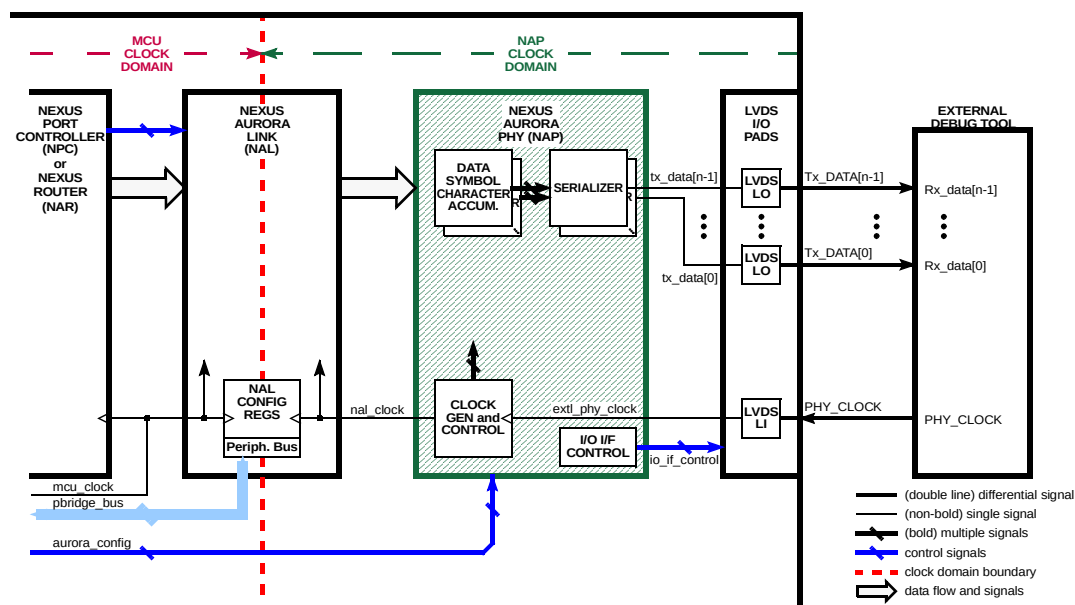
Each NAP lane consists of a data symbol character accumulator which registers the 10-bit character for that lane from the NAL on the rising edge of the NAL clock. Next, a serializer takes the 10-bit character and serializes it into the individual data bits that are then transmitted at the PHY clock rate via the LVDS output drivers to the external debugger receive channel.

Note: The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called Lane Alignment in Aurora terminology), 10b/8b decoding, decode and disparity error tracking in the received data stream.

Figure 2226 shows a high level block diagram of the High Speed Nexus Interface and the NAP.

Figure 2226. High speed Nexus interface block diagram



97.1.1 Features

The features of the NAP are as follows:

- Xilinx Aurora Protocol Specification V2.x compliant
- 1Gb/s per channel trace data payload bandwidth with 1.25 GHz clock
- Simplex mode with static (timer-based) channel training

97.1.2 Modes of operation

The operating mode of the NAP is determined by the device control signals. The functional configuration is determined by the NAR_NAPCR setting; functional operation is enabled by the NAR.

97.1.2.1 Reset

During Nexus Reset all internal NAP resources are reset, all output ports are disabled, and all inputs are ignored. Since there is no device clock domain logic in the NAP itself, the NAP is unaffected by device system reset. Depending on the NAP configuration settings, the NAP may activate upon exiting Nexus Reset if any Aurora lanes are enabled during this reset mode, else it remains disabled and inactive.

97.1.2.2 Disabled/enabled

The NAP is disabled when it comes out of reset if the device is configured for pure device functional mode operation with Nexus trace messaging inhibited. The Aurora LVDS transmit lanes also remain powered down. In addition, any time the NAP block enable is negated the NAP is disabled and the LVDS transmit lanes powered down. Conversely, if the NAP is enabled when it comes out of reset or enabled sometime later after reset is negated, the NAP channels become active and the LVDS output lanes are enabled and can begin to transmit Nexus trace data after the startup time.

97.1.2.3 Link training

Before beginning trace output mode operation, the Aurora Link must go through a training process conducted by the NAL to ensure that an external debugger can correctly reconstruct the transmitted data stream. Training proceeds once the link powers up, and must complete successfully before data transmission can occur.

The NAL initializes and trains the Aurora Link in a simplex configuration using a timer based static-training method. The initialization and training sequence described in Aurora Protocol Specification progresses from one step to the next via the use of programmable timers contained in the NAL. A timer is programmed for each stage of training and as each timer expires, the training sequence proceeds to the next stage until complete. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

97.1.2.4 Transmit

During Nexus trace output operation, it is the responsibility of the NAR to push data into the Aurora Link at a sufficiently high rate to ensure that the link does not become data-starved. The NAL and NAP support up to 4 Transmit lanes. The Aurora encapsulated data is modified by the NAL to ensure the appropriate number of symbols are transmitted through the NAP.

97.2 External signal description

The external signals of the NAP are listed in [Table 2084](#). Each of the NAP signals consists of a positive and negative differential pair.

Table 2084. NAP signals

Signal Name	Direction	Description
PHY_CLOCK (CLKP/CLKN)	Input	NAP clock input from external debug tool
TX Data0 (TX0P/TX0N)	Output	NAP Tx data for lane 0
TX Data1 (TX1P/TX1N)	Output	NAP Tx data for lane 1
TX Data2 (TX2P/TX2N)	Output	NAP Tx data for lane 2
TX Data3 (TX3P/TX3N)	Output	NAP Tx data for lane 3

97.3 Memory map and register definition

There are no control, configuration or status registers within the NAP. All control, configuration and status options for number of lanes and LVDS pads are handled in the NAR_NAPCR which is described in the NAR Chapter.

97.4 Functional description

High-level descriptions of the operation of the NAP and its main functional blocks are given in the following sections.

97.4.1 Data symbol character accumulator

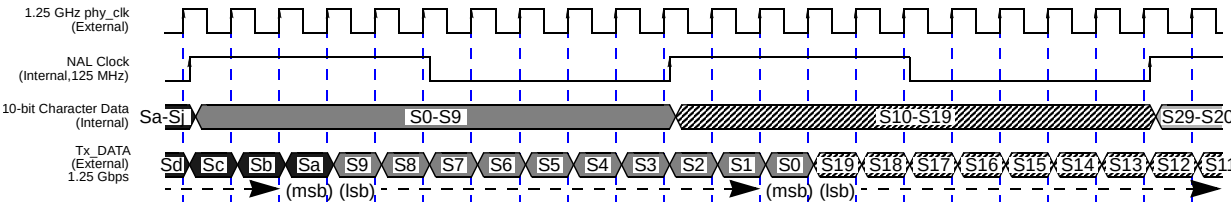
The data symbol character accumulator portion of the NAP for each lane is made up of a symbol character register whose inputs are connected to the 10-bit NAL character bus. The data symbol character accumulator essentially registers a 10-bit character for that lane simultaneous with the rising edge of the NAL clock and presents the 10-bit quantity to the serializer at the appropriate times.

97.4.2 Serializer

The serializer portion of the NAP for each lane takes the data captured by that lane's Data Symbol Character Accumulator and demuxes the data to provide a single serial stream of symbol data to be output by the LVDS LO buffer for that lane at the PHY clock rate.

[Figure 2227](#) shows a high level timing diagram that demonstrates the operation of the serializer for a NAP lane, where S0–S9 represents a 10-bit symbol, S10–S19 represent the next symbol, and so on.

Figure 2227. Serializer timing diagram



97.4.3 Clock generation and control

The Clock Generation and Control block is responsible for generating the necessary clocks and state signals needed by the rest of the NAP. The Data Symbol Character Accumulators and serializers are the main consumers of the signals generated by the Clock Generation and Control block.

97.5 Initialization information

The NAP does not require initialization other than the Nexus Reset and programming of the NAR_NAPCR to configure the number of lanes and LVDS pads.

98 Buddy Device—Data Write Processing Unit (DWPU)

98.1 Introduction

The Data Write Processing Unit (DWPU) supports two specific debug and calibration functions:

- Data trace filtering
- Out of range parameter identification

DWPU takes the Nexus standard messages and processes the data write trace messages to filter out messages based on the tagged addressing schema contained in the Tag RAM. DWPU also generates triggers indicating out of range data values for a set of identified addresses and data ranges.

This figure shows the block diagram of the DWPU identifying various functional block and the data flow.

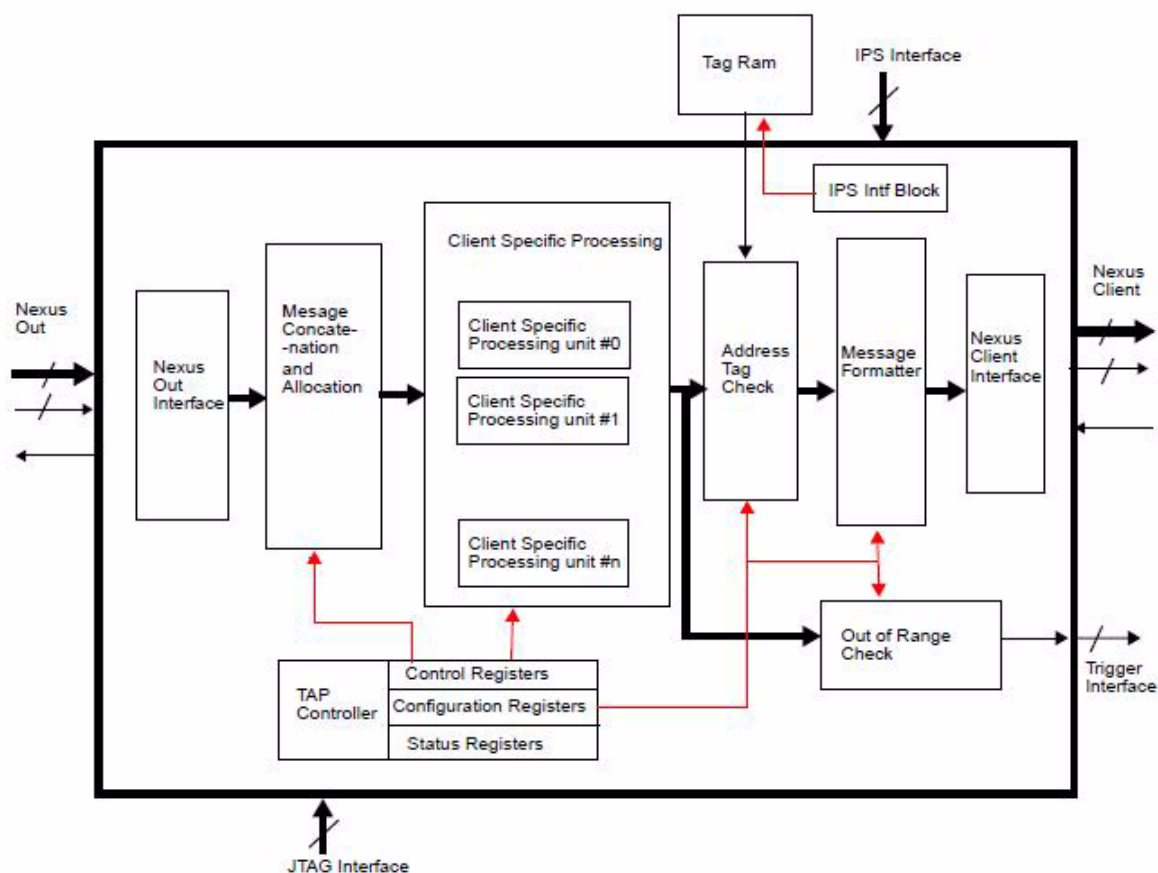


Figure 2228. DWPU block diagram

98.2 Features

DWPU supports the following features:

- Out of range data value detection and data write trace filtering simultaneously from up to five Nexus clients
- Filtering operation that can be individually enabled or disabled for each 64-bit word in system RAM or data memory
- High bandwidth throughput
- Filtered trace stream is returned to the BD NAR to allow routing to the trace port or overlay RAM.
- Detection of out of range data values writes within data write trace stream

98.3 Functional description

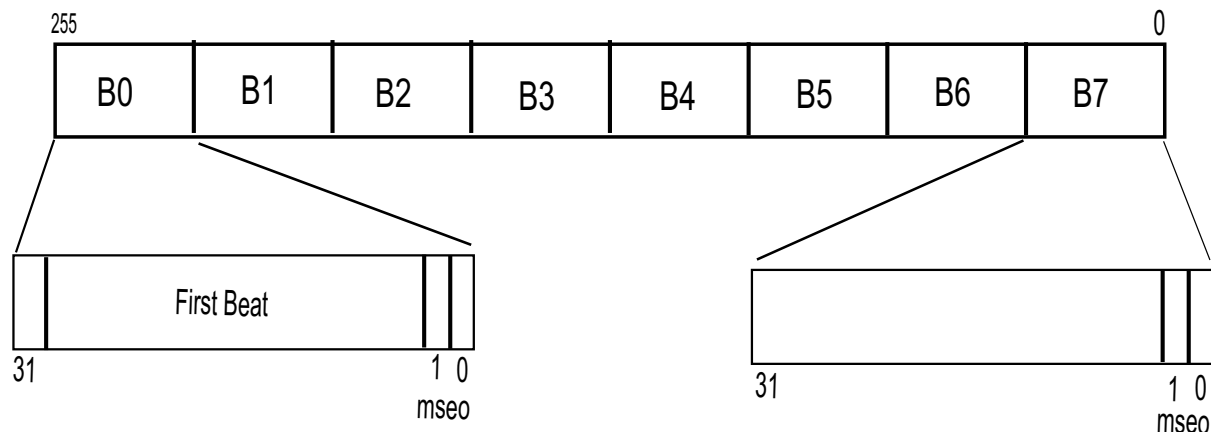
DWPU includes the following functional blocks:

- Nexus Out Interface (NOI)
- Message Concatenation and Allocation (MCA)
- Client Specific Processing (CSP)
- Address Tag Check (ATC)
- Out of Range Check (ORC)
- Message Formatter (MF)
- Nexus Client Interface (NCI)

98.3.1 Nexus Out Interface (NOI)

The Nexus Out Interface (NOI) enables DWPU to receive Nexus message for data write filtering from NAR. NOI maintains a minimum queue of two buffers to accommodate incoming messages. The NAR only initiates a transfer to the DWPU when it has full beats messages. The incoming messages are buffered and then forwarded to the Message Concatenation and Allocation (MCA) for message concatenation and allocation. NOI ensures the temporal order of the incoming messages while forwarding to MCA. NOI also acknowledges the processing of data write messages by MCA. The format for incoming message with NEXUS_WORD = 32 and NUM_BEAT_IN = 8 is shown below.

Figure 2229. Incoming message format



98.3.2 Message Concatenation and Allocation (MCA)

Message Concatenation and Allocation (MCA) starts concatenating message beats once the NOI has a valid input messages to form a complete message. Concatenated messages with selected source IDs contained in within Nexus messages are passed over to appropriate Client Specific Processing Unit (CSPU) for further processing. The following messages are dropped by MCA:

- Messages that do not have a selected source ID
- IDLE messages
- Messages with number of beats larger than defined beats for data-write messages

98.3.3 MCA has the capability of providing one fully concatenated message to a CSPU each clock. **Client Specific Processing (CSP)**

The Client Specific Processing (CSP) block receives the messages from MCA. The received message are processed after extracting all the relevant fields. Different types of message from various clients are received and processed differently based on their type by a specialized unit called the Client Specific Processing Unit (CSPU).

The CSPU receives concatenated data write messages (including synchronization messages) from MCA and provides the following for the Out of Range Check (ORC) and Address Tag Check (ATC) logic to perform filter checks:

- Reconstructed address
- Data value
- Source ID
- Master ID
- Timestamp

CSPU also confirms that each message is a data write message. All other message types are dropped.

There are two CSPU types, one associated with Core Nexus clients and another associated with NXMC clients. Each CSPU simultaneously supports data write messages with up to four different source IDs. Data write with synchronization message is always required to start an address thread for a particular source ID, then each subsequent data write message associated with that source ID contains an encoded address. This table shows the CSPU

type encoding to be used while mapping the source IDs to a CSPU type using SRC_MAP_CTRL.

Table 2085. CSPU encoding

CSTU type	Encoding
Standard	01
NXMC	10

98.3.3.1 Input message formats

DWPU supports various message formats from different source clients. These formats are further processed by the CSPU. The following message formats are supported by the CSPUs, as also described in details the sub-sections:

- Standard data write message format (SDWM)—Includes sync and non-sync type. This is used by most of the clients, i.e., e200 peripheral and e200 main cores.
- AHBMM (modified) data write message format (MDWM)—Includes both sync and non-sync type of messages. Used by NXMC clients only.

98.3.3.1.1 Standard data trace write message format

The standard data trace write messages are of two types, sync and non-sync. The non-sync type contains the data write value and the address of the write access, relative to the previous data trace message. The address and DATA field is considered variable. Data write message information is messaged out in the format shown below. The SRC field values are described in the device-specific chapter that describes how the clients are configured.

1-64 bits	1-32 bits	4 bits	1 bit	4 bits	6 bits
DATA	U-ADDR	DSZ	0	SRC	TCODE (000101)

Figure 2230. Non-sync data trace write message format

The TSTAMP field not shown is an optional fixed field of 26 bits, is inserted optionally by the core clients based on controls received from the BD NAR.

The second type is the DWM synchronization message, whose format is exactly same as above but with full-address (without leading zeros) instead of the relative address. This is to ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DWMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write with synchronization messages is shown below.

1-64 bits	1-32 bits	4 bits	1 bit	4 bits	6 bits
DATA	F-ADDR	DSZ	0	SRC	TCODE (001101)

Figure 2231. Sync data trace write message format

The table below shows the data size (DSZ) encoding for standard data trace message.

Table 2086. Standard DSZ encoding

DSZ	Data size
0000	0-byte
0001	1-byte
0010	2-byte
0011	3-byte
0100	4-byte
0101	5-byte
0110	6-byte
0111	7-byte
1000	8-byte
1001–1111	Reserved

98.3.3.1.2 NXMC (modified) data trace write message format

The NXMC (modified) data trace write sync messages contain the data write value and the address of the write access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encoding to either 8, 16, 32 or 64. The TSTAMP field is 26 bits. Data write message information is messaged out in the format shown below. The SRC field values are described in the device-specific chapter that describes how the clients are configured.

8/16/32/64 bits	1-32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010)

Figure 2232. Non-sync M-Data data trace write message format

The TSTAMP field not shown over here is an optional fixed field of 26 bits, is inserted optionally by the AHBMM clients based on controls received from NAR.

The second type is the MDWM synchronization message, whose format is exactly same as above but with full-address (without leading zeros) instead of the relative address. This is to ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent MDWMs, in which only the unique

portion of the data trace address is transmitted. The format for data trace write with synchronization messages is shown below.

8/16/32/64 bits	1-32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100)

Figure 2233. Sync M-Data data trace write message format

The table below shows the data size (DSZ) encoding for NXMC data trace message.

Table 2087. NXMC DSZ encoding

DSZ	Data size
001	1-byte
010	2-byte
011	3-byte
100	4-byte
000,101–111	Reserved

98.3.4 Address Tag Check (ATC)

DWPU allows filtering operation on data writes to any address within system RAM and tightly coupled data memory. Each 64-bit word in the supported system RAM or data memory address space corresponds to a single bit in Tag RAM; therefore, the required Tag RAM size is $1/64^{\text{th}}$ of the address space to be filtered. Data write messages within the supported address range are dropped by ATC if the corresponding bit in Tag RAM is not set. The data write messages with the corresponding Tag bit set are passed over to Message Formatter for further processing.

Writes to addresses outside of the supported address range can be configured to be either removed from the trace stream or retained. This option is controlled by a single user accessible control bit. Data reads and other Nexus message formats submitted in the DWPU trace stream are always dropped by DWPU. Tag RAM is accessible through software (IPS) as well as JTAG interface via PD addressing. All the IPS access is possible in supervisor mode only. The access to TAG RAM is carried out indirectly through three registers: TRC_CTRL, TRC_ADDR and TRC_DATA registers. Users should write the TRC_CTRL register followed by the TRC_ADDR register before accessing the TRC_DATA register. Any write access to the TRC_ADDR register results in a Tag RAM access request. Users should ensure that the Tag RAM request is completed by providing the required TRC_DATA access based on the length of the transfer. In case of block access, the TRC_DATA register is accessed subsequently and users should ensure that the block access length does not cross the region boundary. Deviation in the sequence of register access for TAG RAM access results in incorrect and incomplete transfers.

In PD addressing, the configured PD address in the TRC_ADR register is translated to a TAG RAM address using mapping derived from the FAR_SA_CFGn and FAR_RS_CFGn register. The PD address configured in the FAR_SA_CFG0 register is always translated to the zeroth address of Tag RAM.

The equation to convert system/PD address to Tag RAM address where the System/PD address falls within region n is given below:

$$\text{Tag RAM address} = \text{System/PD address}[31:8] - \text{Region n base address}[31:8] + \text{Total sum of previous region (n-1) size}[31:8]$$

The Tag RAM address for System/PD address that falls on region 2 is given by:

$$\text{System/PD address}[31:8] - \text{Region 2 base address}[31:8] + (\text{region 0 size}[31:8] + \text{region 1 size}[31:8])$$

This figure shows an example of the Tag RAM organization for 8 Kbyte Tag RAM with PD base address configured.

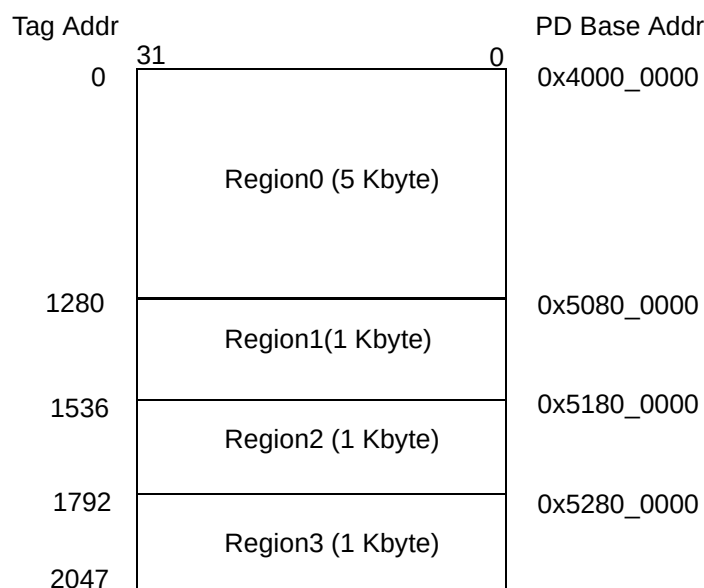


Figure 2234. Tag RAM organization example

The table below shows an example of address translation from PD addresses to Tag RAM address.

Table 2088. Address translation example

PD address		Tag RAM address
Start	End	
0x4000_0000	0x4000_00FF	0x0
0x4000_0100	0x4000_01FF	0x1
0x4004_FF00	0x4004_FFFF	0x4FF
0x5080_0000	0x5080_00FF	0x500
0x5180_0000	0x5180_00FF	0x600
0x5280_0000	0x5280_00FF	0x700
0x5280_FF00	0x5280_FFFF	0x7FF

During the data write filtering operation, the system RAM or data memory address is extracted from the input message. It is then compared to ascertain the address space (whether it is within system RAM space or data memory or outside the supported address range). Finally it is translated to Tag RAM address and initiate a read to Tag RAM. From the Tag RAM 32-bit read accessed, the system/PD address bit [7:3] determine the Tag bit position that is associated with the decoded absolute system/PD address. If the corresponding bit is set, then the input message is passed for message formatting else it is dropped by ATC.

This figure shows the data flow for address Tag filtering.

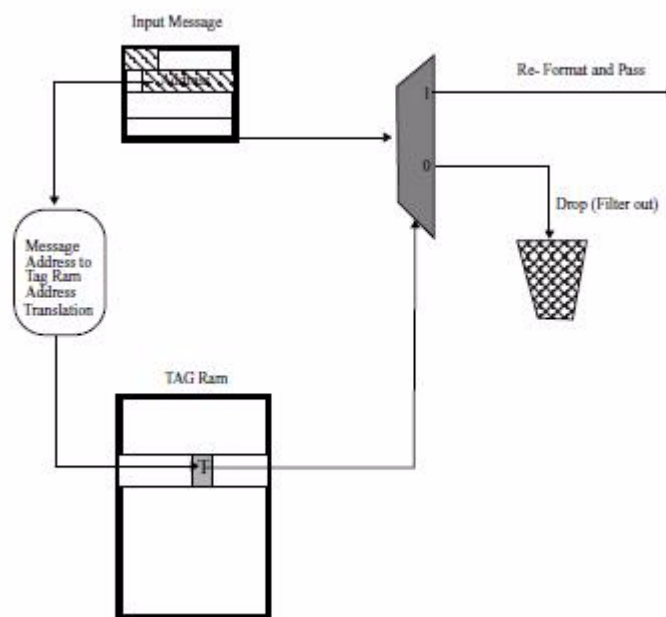


Figure 2235. Address tag filtering

98.3.5 Out of Range Check (ORC)

In addition to the trace filtering function, the DWPU also supports detection of outside/inside range writes to RAM locations. Both the data range values and address of RAM locations to be checked are user specified. It also has configurable support to detect target inside or outside data range values specified.

A total of four sets of address and data range comparators are supported, where each set of comparators include an address match comparator, a greater than or equal data value comparator and a less than or equal data value comparator. This figure shows the block diagram for range detect comparators.

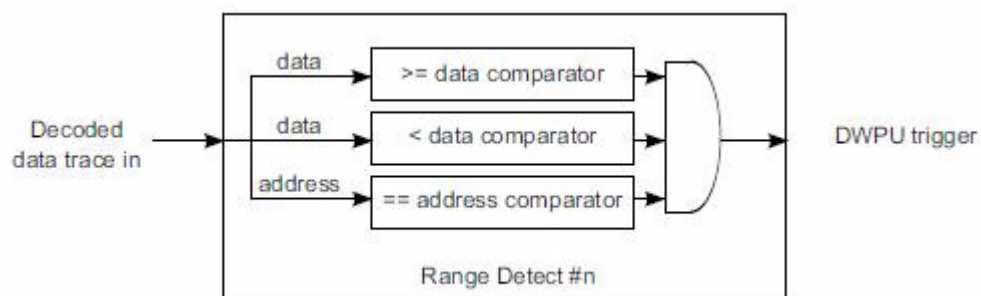


Figure 2236. Range detect comparators

ORC supports unsigned, signed and floating point data ranges. Signed numbers are represented in 2's complement. 32-bit IEEE-754 single-precision floating-point format is used for floating point representation.

ORC has a configurable option to support for 8/16/32 bit unsigned/signed range detection. Each data comparator supports bit masking and configurable pairing of comparators. Address comparison is carried out with full the device address. An output signal from DWPU is triggered in response to a detected out of range condition along with a status bit being set in a ORC status register indicating which range comparator has caused the trigger.

98.3.5.1 Comparator capabilities

The primary usage for out of range detect function is to detect out of range conditions on unsigned integer and pointer variables, although it also supports signed and floating point data types comparison. It also has a configurable option to support for 8/16/32 bit unsigned range detection. The unsigned data is directly compared with the stored data ranges.

To support use with signed variables, each data comparator supports signed operation. All the signed numbers are represented in two's complement. Data comparison happens only when the sign bit matches.

To support range of floating point variables, each data comparator also supports bit masking and configurable pairing of comparators to generate a combined watch point trigger enabling individual comparators to compare the exponent and significand.

The pairing logic is shown in this figure.

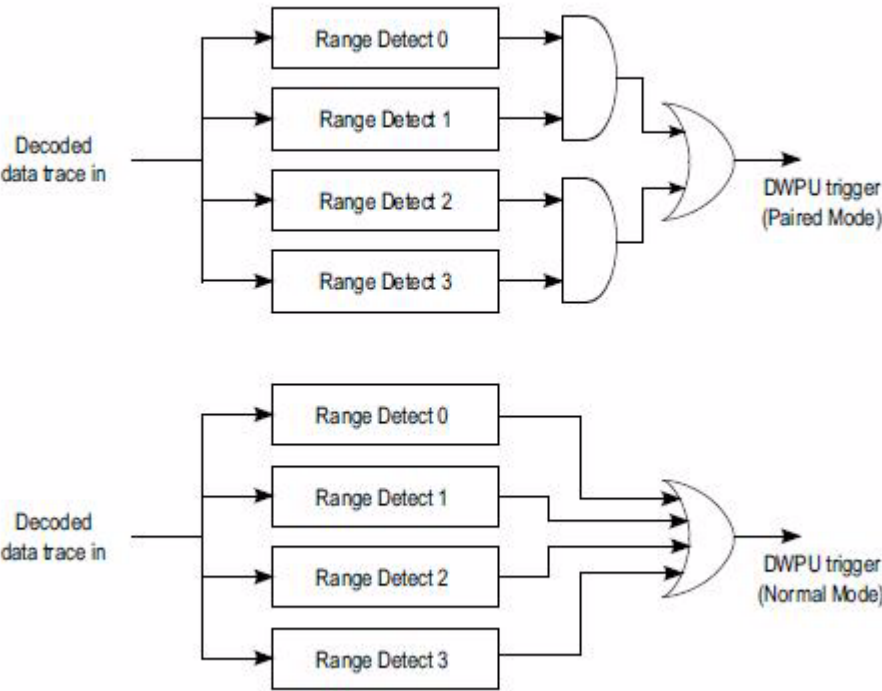


Figure 2237. DWPU trigger mode

98.3.6 Message Formatter (MF)

Message Formatter (MF) module receives message information from ATC and reformats the data write messages to comply with output Nexus message format. A user defined message format is used for sending out the filtered messages as shown below.

8/16/32/64 bits	32 bits	4 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111110)

Figure 2238. DWPU message format

Table 2089. Message field description

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
DWPU message	6	6	TCODE	fixed	TCODE number = 62
	4	4	SRC	fixed	Source client identifier
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access. For core clients 1111 is inserted. For the NXMC clients, same value is inserted as received.
	4	4	DSZ	fixed	Data size
	32	32	F-ADDR	fixed	Full data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	26	26	TSTAMP	fixed	Timestamp (Optional and omitted if there is no valid timestamp)

98.3.7 Nexus Client Interface (NCI)

The Nexus Client Interface (NCI) sends out the formatted filtered data write messages to BD NAR. It also supports the buffer reallocation to increase the Q size.

98.3.8 IEEE 1149.1 (JTAG) Test Access Port

The DWPU module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The DWPU module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in [Section 98.3.11, Register access via JTAG](#).

98.3.9 Enabling DWPU operation

The DWPU module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The DWPU module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The DWPU module is also disabled if a Power-on-Reset (POR) event occurs.

If the DWPU module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

98.3.10 Enabling TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the DWPU TAP is accessed, this information helps the development tool obtain information about the DWPU module it is accessing, such as version, sequence, feature set, etc.

98.3.11 Register access via JTAG

Access to DWPU register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all DWPU registers according to the map in [Table 2093](#).

Reading/writing of a DWPU register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the DWPU register to be accessed by providing an index (see [Table 2093](#)), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown below.

Figure 2239. JTAG DR for DWPU register access

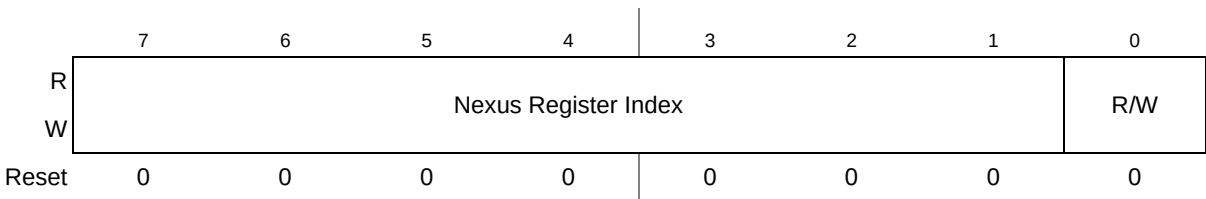


Table 2090. JTAG DR field description for DWPU register access

Field	Description
DWPU register index	Selected from values in Table 2093
Read/Write (R/W)	0Read 1Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
 - a) During a read access, data is latched from the selected DWPU register when the JTAG state machine passes through the Capture-DR state.
 - b) During a write access, data is latched into the selected DWPU register when the JTAG state machine passes through the Update-DR state.

98.4 Configuration

DWPU has a set of registers and Tag RAM which has to be configured and initialized before enabling DWPU. The following sequence must be followed for configuring DWPU:

- Configure the GLB_CTRL register timestamp field (GLB_CTRL[TSC]) and data write outside the identified RAM areas (GLB_CTRL[ORC]) control.
- Configure the region enable GLB_CTRL[FAREn]. Ensure that there is no disabled region between two enabled region and regions are enabled such that they start from region 0 and increase by region number.
- Configure FAR_SA_CFGn. Start address for each region is aligned to 256-byte boundaries.
- Configure FAR_RS_CFGn. Region size is an integral multiple of 256-bytes.
- Initialize the Tag RAM for the total region configured
- Ensure all the Tag access request are completed without error by examining the TRC_CTRL[11:8] bit for JTAG access and TRC_CTRL[10:8] for IPS access
- Configure SRC_MAP_CTLRn register for all the sources.
- Configure ORC_CTRLn register and the data ranges and addresses.
- Enable DWPU by setting the GLB_CTRL[MEN] bit.

98.4.1 Tag RAM initialization

DWPU has a 32-bit Tag RAM for filtering data write traces with depth being configurable. Each bit of Tag RAM corresponds to 8 bytes of system RAM. The Tag RAM supports bit masking, enabling the software to configure at least single bit of Tag RAM. Tag RAM can be organized into four different regions with configurable base addresses and region sizes.

Tag RAM can be initialized either by JTAG or IPS interface through indirect access. In indirect access, the TRC_CTRL register is configured first followed by the TRC_ADDR register. Subsequently, write/read access to the TRC_DATA register results in Tag RAM write or read access respectively. Consider an 8 Kbyte Tag RAM, organized into four regions with the following start address and region size configured.

Table 2091. Tag RAM configuration

Region	FAR_SA_CFGn	FAR_RS_CFGn
0	0x4000_0000	0x0005_0000
1	0x5080_0000	0x0001_0000
2	0x5180_0000	0x0001_0000
3	0x5280_0000	0x0001_0000

The following table provides an overview for Tag RAM address translation and write data mask generation based on the TRC_CTRL[DSM] and TRC_ADDR registers for the above Tag RAM organization.

Table 2092. System address to Tag address translation

DSM	TRC_ADDR	Region	Tag address	Write mask	Comment
000	0x4000_0000	0	0x000	0x0000_0000	32-bit write
001	0x4000_0000	0	0x000	0xFFFF_0000	LSB 16-bit write
001	0x4000_0080	0	0x000	0x0000_FFFF	MSB 16-bit write
100	0x4004_FF08	0	0x4FF	0xFFFF_FFFD	Single-bit write
100	0x4005_0000	None	—	—	Outside the supported address range
010	0x5080_0000	1	0x500	0xFFFF_FF00	8-bit write
010	0x5080_0040	1	0x500	0xFFFF_00FF	8-bit write
010	0x5080_0180	1	0x501	0xFF00_FFFF	8-bit write
100	0x5180_FFF8	2	0x6FF	0x7FFF_FFFF	Single-bit write
000	0x5280_0000	3	0x700	0x000_0000	32-bit write
000	0x5280_0180	3	0x701	—	Address unaligned with DSM; Result in incorrect write to Tag RAM

98.5 Register definition

This section describes the DWPU Nexus register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. In addition, some register are accessible through the IPS interface.

The table below shows the JTAG mapped DWPU registers.

Table 2093. JTAG DWPU registers

Nexus register index	Register	Access	Read address	Write address	Reset value
0x02	Global Control (GLB_CTRL)	R/W	0x04	0x05	0x0000_0002
0x03	Source Mapping Control 0 (SRC_MAP_CTLR0)	R/W	0x06	0x07	0x0000_0000
0x04	Source Mapping Control 1 (SRC_MAP_CTLR1)	R/W	0x08	0x09	0x0000_0000
0x05	Source Mapping Control 2 (SRC_MAP_CTLR2)	R/W	0x0A	0x0B	0x0000_0000
0x0A	Filter Address Range Start Address 0 (FAR_SA_CFG0)	R/W	0x14	0x15	0x0000_0000
0x0B	Filter Address Range Region Size 0 (FAR_RS_CFG0)	R/W	0x16	0x17	0x0000_0000
0x0C	Filter Address Range Start Address 1 (FAR_SA_CFG1)	R/W	0x18	0x19	0x0000_0000
0x0D	Filter Address Range Region Size 1 (FAR_RS_CFG1)	R/W	0x1A	0x1B	0x0000_0000
0x0E	Filter Address Range Start Address 2 (FAR_SA_CFG2)	R/W	0x1C	0x1D	0x0000_0000
0x0F	Filter Address Range Region Size 2 (FAR_RS_CFG2)	R/W	0x1E	0x1F	0x0000_0000
0x10	Filter Address Range Start Address 3 (FAR_SA_CFG3)	R/W	0x20	0x21	0x0000_0000
0x11	Filter Address Range Region Size 3 (FAR_RS_CFG3)	R/W	0x22	0x23	0x0000_0000

Table 2093. JTAG DWPU registers (continued)

Nexus register index	Register	Access	Read address	Write address	Reset value
0x1A	Tag RAM Control (TRC_CTRL)	R/W	0x34	0x35	0x0000_0000
0x1B	Tag RAM Configuration Address (TRC_ADR)	R/W	0x36	0x37	0x0000_0000
0x1C	Tag RAM Configuration Data (TRC_DATA)	R/W	0x38	0x39	0x0000_0000
0x20	Out of Range Check Control 1 (ORC_CTRL1)	R/W	0x40	0x41	0x0000_0000
0x21	Out of Range Check Control 2 (ORC_CTRL2)	R/W	0x42	0x43	0x0000_0000
0x22	Global Status (GLB_STAT)	R	0x44	0x45	0x0004_0000
0x24	Out of Range Configuration Address 0 (ORC_CFG0_ADR)	R/W	0x48	0x49	0x0000_0000
0x25	Out of Range Configuration Data High 0 (ORC_CFG0_DH)	R/W	0x4A	0x4B	0x0000_0000
0x26	Out of Range Configuration Data Low 0 (ORC_CFG0_DL)	R/W	0x4C	0x4D	0x0000_0000
0x27	Out of Range Configuration Data Mask 0 (ORC_CFG0_DM)	R/W	0x4E	0x4F	0x0000_0000
0x28	Out of Range Configuration Address 1 (ORC_CFG1_ADR)	R/W	0x50	0x51	0x0000_0000
0x29	Out of Range Configuration Data High 1 (ORC_CFG1_DH)	R/W	0x52	0x53	0x0000_0000
0x2A	Out of Range Configuration Data Low 1 (ORC_CFG1_DL)	R/W	0x54	0x55	0x0000_0000
0x2B	Out of Range Configuration Data Mask 1 (ORC_CFG1_DM)	R/W	0x56	0x57	0x0000_0000
0x2C	Out of Range Configuration Address 2 (ORC_CFG2_ADR)	R/W	0x58	0x59	0x0000_0000
0x2D	Out of Range Configuration Data High 2 (ORC_CFG2_DH)	R/W	0x5A	0x5B	0x0000_0000
0x2E	Out of Range Configuration Data Low 2 (ORC_CFG2_DL)	R/W	0x5C	0x5D	0x0000_0000
0x2F	Out of Range Configuration Data Mask 2 (ORC_CFG2_DM)	R/W	0x5E	0x5F	0x0000_0000
0x30	Out of Range Configuration Address 3 (ORC_CFG3_ADR)	R/W	0x60	0x61	0x0000_0000
0x31	Out of Range Configuration Data High 3 (ORC_CFG3_DH)	R/W	0x62	0x63	0x0000_0000
0x32	Out of Range Configuration Data Low 3 (ORC_CFG3_DL)	R/W	0x64	0x65	0x0000_0000
0x33	Out of Range Configuration Data Mask 3 (ORC_CFG3_DM)	R/W	0x66	0x67	0x0000_0000

The table below shows the IPS mapped DWPU registers.

Table 2094. Memory mapped DWPU registers

Offset	Registers	Access	Reset value
0x00	Tag RAM Control (TRC_CTRL)	R/W	0x0000_0000
0x04	Tag RAM Configuration Address (TRC_ADR)	R/W	0x0000_0000
0x08	Tag RAM Configuration Data (TRC_DATA)	R/W	0x0000_0000

98.5.1 Register descriptions

98.5.1.1 Global Control (GLB_CTRL)

The GLB_CTRL register is shown in this figure.

Figure 2240. GLB_CTRL register

Index 0x02

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F	F	F	F		0	0	0	0	0			
W																				R	R	R	R	TSC						ORC	SFT_RST	MEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 2095. GLB_CTRL field descriptions

Field	Description
MEN	Module Enable. 0 DWPU disabled 1 DWPU enabled
SFT_RST	Active low soft reset. 0 Enabled soft reset 1 Disable soft reset
ORC	0 Data writes outside the identified RAM areas are flushed 1 Data writes outside the identified RAM areas are passed
TSC	0 Pass incoming timestamp and include in output message format (requires more trace bandwidth or overlay RAM locations) 1 Drop timestamp from incoming messages
FARE[0–3]	0 Filter Address Range n Configuration (FAR_SA_CFGn, FAR_RS_CFGn) disabled. 1 Filter Address Range n Configuration (FAR_SA_CFGn, FAR_RS_CFGn) enabled.

98.5.1.2 Source Mapping Control n (SRC_MAP_CTLRn)

The SRC_MAP_CTLRn register is shown in this figure.

Figure 2241. SRC_MAP_CTLRn register

Index:

n=0 0x03

n=1 0x04

n=2 0x05

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN1	0	0	0	0	CSP_TYPE1			SRC_ID1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EN0	0	0	0	0	CSP_TYPE0			SRC_ID0							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 2096. SRC_MAP_CTLRn field descriptions

Field	Description
SRC_ID0	Source ID selected
CSP_TYPE0	CSPU Type to which the SRC_ID0 is mapped 000 Unmapped (to be dropped by CSP) 001 Standard message 010 NXMC message 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
EN0	0 SRC_ID0 field is not used or mapped to a CSPU. Messages with unmapped source IDs are flushed 1 SRC_ID and CSP_TYPE0 are used for selecting and mapping data write messages
SRC_ID1	Source ID selected

Table 2096. SRC_MAP_CTLRn field descriptions (continued)

Field	Description
CSP_TYPE1	CSPU Type to which the SRC_ID1 is mapped 000 Unmapped (to be dropped by CSP) 001 Standard message 010 NXMC message 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
EN1	0 SRC_ID0 field is not used or mapped to a CSPU. Messages with unmapped source IDs are flushed 1 SRC_ID and CSP_TYPE0 are used for selecting and mapping data write messages

98.5.1.3 Filter Address Range Start Address n (FAR_SA_CFGn)

The FAR_SA_CFGn register is shown in this figure.

Figure 2242. FAR_SA_CFGn register

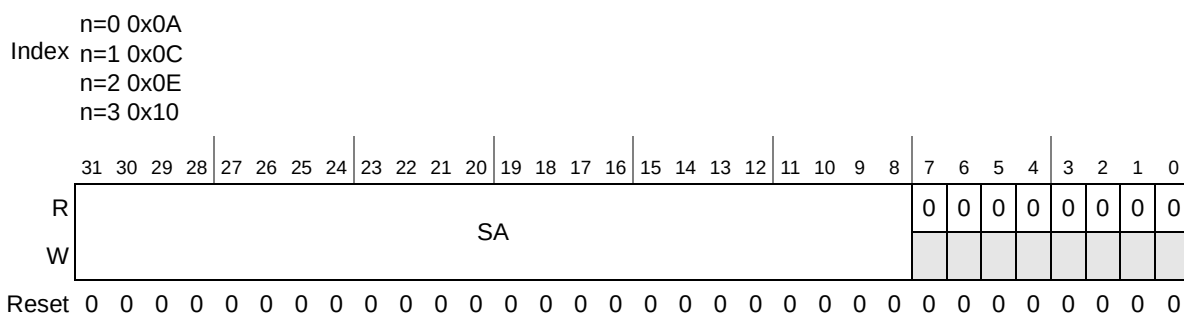


Table 2097. FAR_SA_CFGn field descriptions

Field	Description
SA	Start PD address for the region, should be at 265-byte boundaries

98.5.1.4 Filter Address Range Region Size n (FAR_RS_CFGn)

The FAR_RS_CFGn register is shown in this figure.

Figure 2243. FAR_RS_CFGn register

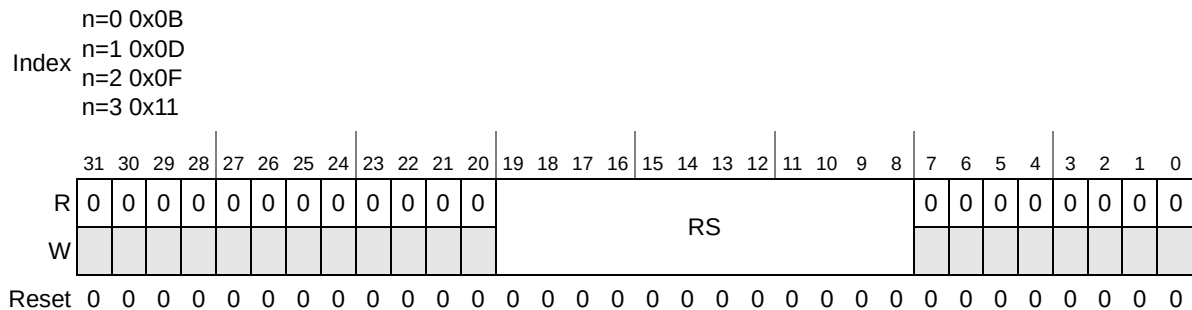


Table 2098. FAR_RS_CFGn field descriptions

Field	Description
RS	Region size in 256-bytes

98.5.1.5 Tag RAM Control (TRC_CTRL)

The TRC_CTRL register is shown in this figure.

Figure 2244. TRC_CTRL register

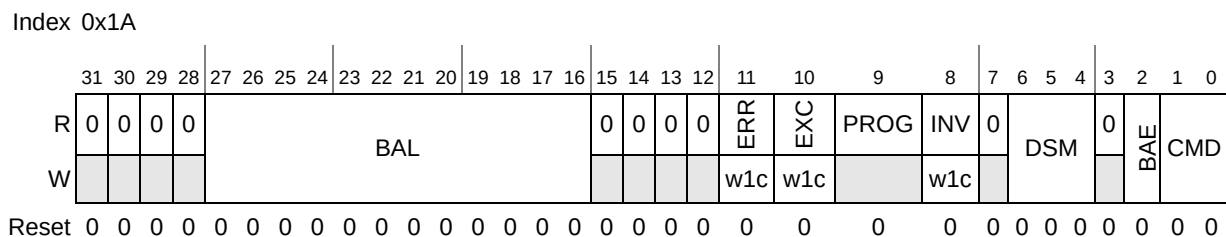


Table 2099. TRC_CTRL field descriptions

Field	Description
CMD	00 No read or write access to Tag RAM 01 Read access to Tag RAM. Any write to TRC_ADR register results in read access to the Tag RAM. Read data is stored in the TRC_DATA register. 10 Write access to Tag RAM. Any write to TRC_ADR register results in write access to the Tag RAM. Valid write data needs to be first written into the TRC_DATA register. 11 Reserved
BAE	Block Access Enable. Block access is carried out with 32-bit Data Size Mode only. 0 Block Access disable 1 Block Access enable

Table 2099. TRC_CTRL field descriptions (continued)

Field	Description
DSM	Data Size Mode. Valid only for writes, all reads are fixed 32 bits only. 000 Word (32 bit) Access to the Tag RAM. 001 Half-Word (16-bit) access to Tag RAM. 16-bit LSBs of the TRC_DATA is used. 010 Byte (8-bit) access to the Tag RAM. 8-bit LSBs of the TRC_DATA is used. 100 Bit Access to the Tag RAM. LS bit of the TRC_DATA is used.
INV	Invalid JTAG access. Generated when the targeted PD address falls outside the configured Tag RAM region 0 No error 1 Invalid address
PROG	JTAG access to Tag RAM in progress 0 No Tag RAM access 1 JTAG Tag RAM access
EXC	Excess JTAG access. Generated when JTAG access to Tag RAM excess the configured access length 0 No excess access 1 Excess access
ERR	Error JTAG access. Generated when read/write access to Tag RAM before getting the grant. Also generated on new JTAG Tag RAM access request before completing the previous access. <i>Note: This field is not present in the IPS mapped version of this register.</i> 0 No Error 1 Error
BAL	Block Access Length. Valid only when BAE = 1

98.5.1.6 Tag RAM Configuration Address (TRC_ADR)

The TRC ADR register is shown in this figure.

Figure 2245. TRC ADR register

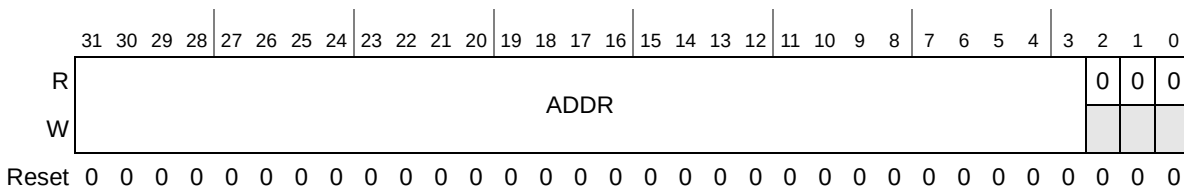


Table 2100. TRC_ADR field descriptions

Field	Description
ADDR	PD Address. It should be aligned depending on Data Size mode field. DSM = 000, 256 byte aligned DSM = 001, 128 byte aligned DSM = 010, 64 byte aligned DSM = 100, 8 byte aligned (default) <i>Note: If the PD address is not aligned to the DSM, this results in a erroneous data read/write.</i>

98.5.1.7 Tag RAM Configuration Data (TRC_DATA)

The TRC_DATA register is shown in this figure.

Figure 2246. TRC_DATA register

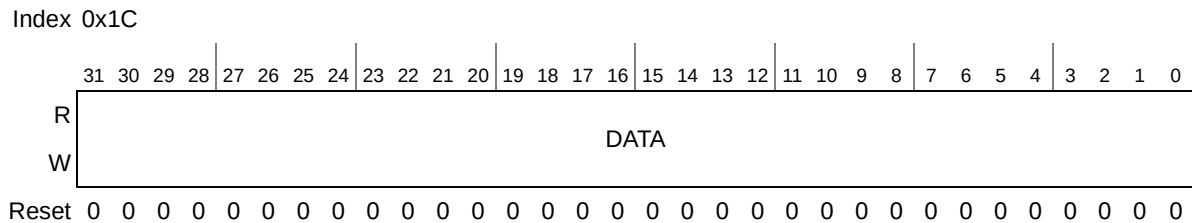


Table 2101. TRC_DATA field descriptions

Field	Description
DATA	Tag RAM Data

98.5.1.8 Out of Range Check Control 1 (ORC_CTRL1)

The ORC_CTRL1 register is shown in this figure.

Figure 2247. ORC_CTRL1 register

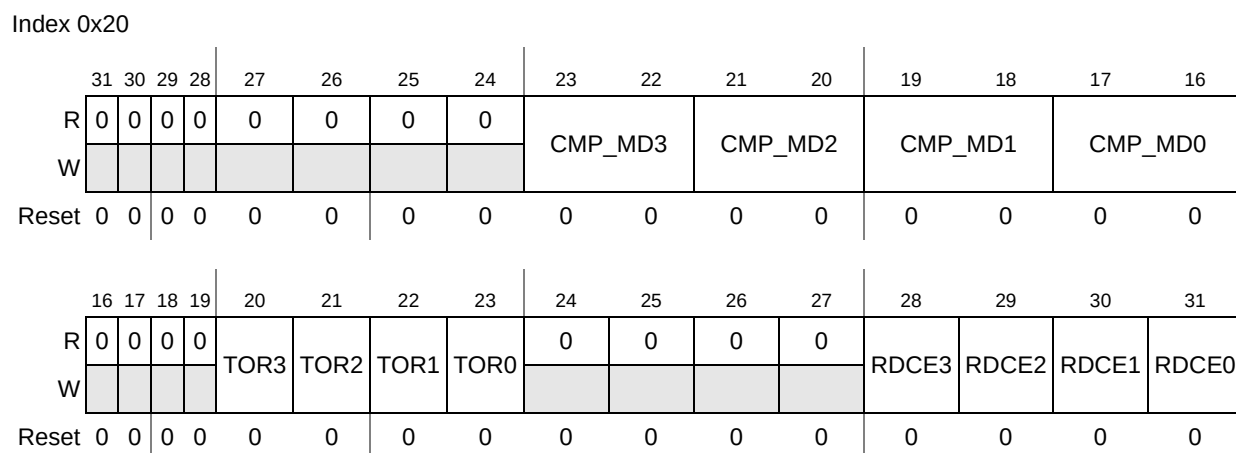


Table 2102. ORC_CTRL1 field descriptions

Field	Description
RDCE0	0 Disable Range Detect comparator 0 1 Enable Range Detect comparator 0
RDCE1	0 Disable Range Detect comparator 1 1 Enable Range Detect comparator 1
RDCE2	0 Disable Range Detect comparator 2 1 Enable Range Detect comparator 2
RDCE3	0 Disable Range Detect comparator 3 1 Enable Range Detect comparator 3

Table 2102. ORC_CTRL1 field descriptions (continued)

Field	Description
TOR0	Comparison Mode for Comparator 0 0 Trigger is generated if data write value is within the range $\text{ORC_CFG0_DL} \leq \text{data_wr_value} \leq \text{ORC_CFG0_DH}$ 1 Trigger is generated if data write value is outside the range $\text{data_wr_value} < \text{ORC_CFG0_DL}$ or $\text{data_wr_value} > \text{ORC_CFG0_DH}$
TOR1	Comparison Mode for Comparator 1 0 Trigger is generated if data write value is within the range $\text{ORC_CFG1_DL} \leq \text{data_wr_value} \leq \text{ORC_CFG1_DH}$ 1 Trigger is generated if data write value is outside the range $\text{data_wr_value} < \text{ORC_CFG1_DL}$ or $\text{data_wr_value} > \text{ORC_CFG1_DH}$
TOR2	Comparison Mode for Comparator 2 0 Trigger is generated if data write value is within the range $\text{ORC_CFG2_DL} \leq \text{data_wr_value} \leq \text{ORC_CFG2_DH}$ 1 Trigger is generated if data write value is outside the range $\text{data_wr_value} < \text{ORC_CFG2_DL}$ or $\text{data_wr_value} > \text{ORC_CFG2_DH}$
TOR3	Comparison Mode for Comparator 3 0 Trigger is generated if data write value is within the range $\text{ORC_CFG3_DL} \leq \text{data_wr_value} \leq \text{ORC_CFG3_DH}$ 1 Trigger is generated if data write value is outside the range $\text{data_wr_value} < \text{ORC_CFG3_DL}$ or $\text{data_wr_value} > \text{ORC_CFG3_DH}$
CMP_MD0	00 8 bit data comparison for comparator 0. 01 16 bit data comparison for comparator 0. 10 32 bit data comparison for comparator 0. 11 Reserved <i>Note: All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.</i>
CMP_MD1	00 8 bit data comparison mode for comparator 1. 01 16 bit data comparison mode for comparator 1. 10 32 bit data comparison mode for comparator 1. 11 Reserved <i>Note: All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.</i>
CMP_MD2	00 8 bit data comparison mode for comparator 2. 01 16 bit data comparison mode for comparator 2. 10 32 bit data comparison mode for comparator 2. 11 Reserved <i>Note: All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.</i>
CMP_MD3	00 8 bit data comparison mode for comparator 3. 01 16 bit data comparison mode for comparator 3. 10 32 bit data comparison mode for comparator 3. 11 Reserved <i>Note: All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.</i>

98.5.1.9 Out of Range Check Control 2 (ORC_CTRL2)

The ORC_CTRL2 register is shown in this figure.

Figure 2248. ORC_CTRL2 register

Index 0x21

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	DTYP3		DTYP2		DTYP1		DTYP0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	GTM	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2103. ORC_CTRL2 field descriptions

Field	Description
GTM	Generate Trigger on Mismatch 0 Disable trigger generation on data-size mismatch 1 Enable trigger generation on data-size mismatch <i>Note: Mismatch happens if the data write size is less than comparator size (CMP_MD)</i>
DTYP0	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
DTYP1	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
DTYP2	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
DTYP3	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved

98.5.1.10 Global Status (GLB_STAT)

The GLB_STAT register is shown in this figure.

Figure 2249. GLB_STAT register

Index 0x22

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIDLE	0	TRSE
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	DSM3	DSM2	DSM1	DSM0	0	0	0	0	TRDC3	TRDC2	TRDC1	TRDC0
W					w1c	w1c	w1c	w1c					w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2104. GLB_STAT field descriptions

Field	Description
DIDLE	DWPU module idle indication 1 Indicates that there is no more processing going on inside the module, or the all the messages have been completely processed and sent out.
TRSE	Tag RAM Size Error 1 Configured PD region for ATC exceeds the maximum Tag RAM size.
TRDC0	1 Data range comparison for ORC_CFG0_ADR was successful
TRDC1	1 Data range comparison for ORC_CFG1_ADR was successful
TRDC2	1 Data range comparison for ORC_CFG2_ADR was successful
TRDC3	1 Data range comparison for ORC_CFG3_ADR was successful
DSM0	1 Received data write trace size is smaller than comparator size (CMP_MP0)
DSM1	1 Received data write trace size is smaller than comparator size (CMP_MP1)
DSM2	1 Received data write trace size is smaller than comparator size (CMP_MP2)
DSM3	1 Received data write trace size is smaller than comparator size (CMP_MP3)

98.5.1.11 Out of Range Configuration Address n (ORC_CFGn_ADR)

The ORC_CFGn_ADR register is shown in this figure.

Figure 2250. ORC_CFGn_ADR register

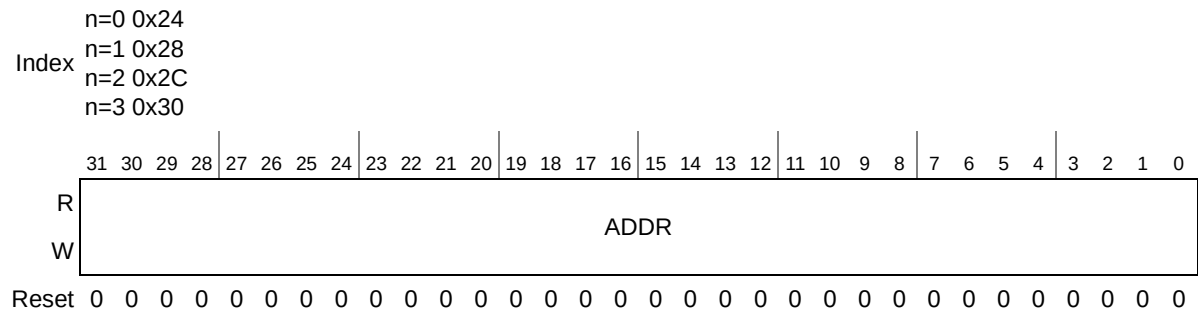


Table 2105. ORC_CFGn_ADR field descriptions

Field	Description
ADDR	Address value for the Data range comparison.

98.5.1.12 Out of Range Configuration Data High n (ORC_CFGn_DH)

The ORC_CFGn_DH register is shown in this figure.

Figure 2251. ORC_CFGn_DH register

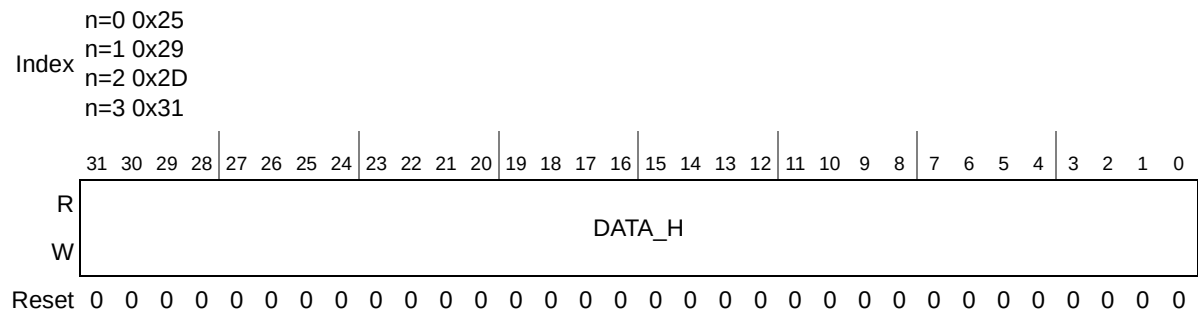


Table 2106. ORC_CFGn_DH field descriptions

Field	Description
DATA_H	Data High value for the comparator.

98.5.1.13 Out of Range Configuration Data Low n (ORC_CFGn_DL)

The ORC_CFGn_DL register is shown in this figure.

Figure 2252. ORC_CFGn_DL register

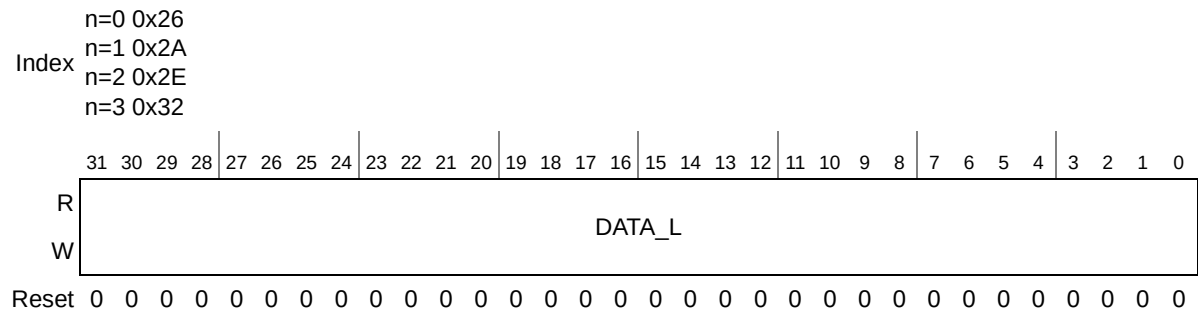


Table 2107. ORC_CFGn_DL field descriptions

Field	Description
DATA_L	Data low value for the comparator.

98.5.1.14 Out of Range Configuration Data Mask n (ORC_CFGn_DM)

The ORC_CFGn_DM register is shown in this figure.

Figure 2253. ORC_CFGn_DM register

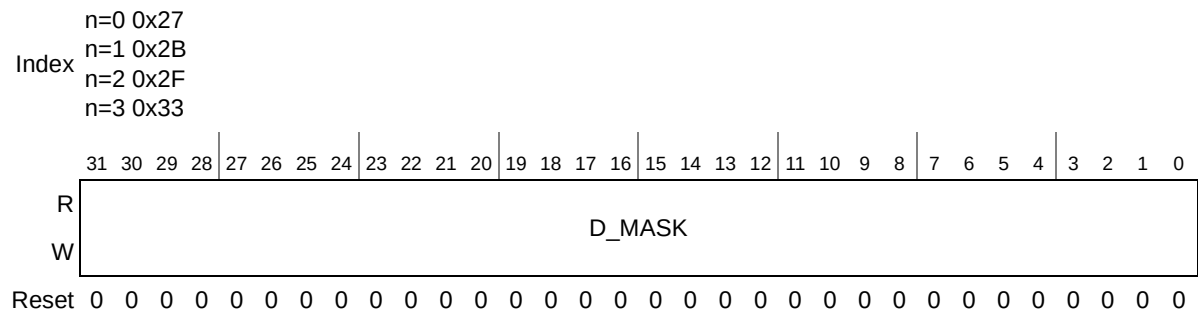


Table 2108. ORC_CFGn_DM field descriptions

Field	Description
D_MASK	Data Mask value for the comparator (bit wise). '0' Data comparison not masked '1' Data comparison masked

99 Buddy Device—Nexus Read Write Access (RWA)

The Nexus Read Write Access (RWA) module allows an external tool connected to the Emulation and Debug Device (ED) to read or write the Buddy Die calibration and trace memory via the JTAG interface. All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). The RWA module supports single as well as block reads and writes.

The RWA module is enabled by loading a single instruction (ACCESS_AUX_RWA_BD = 0x23) into the JTAG Instruction Register (IR). After the RWA module has been enabled, access to the RWA registers is enabled by loading an instruction (CORE_ENABLE) into the JTAG IR, as described in [Section 99.2, Nexus register access via JTAG](#). The RWA module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. The RWA module is also disabled if a Power-on-Reset (POR) event occurs. When disabled, the RWA module registers are not available for (32-bit or less) reads or writes.

99.1 Register definition

[Table 2109](#) details the register map for the RWA module.

Table 2109. RWA register map

Register	Nexus access opcode	Read/write	Read address	Write address
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15

99.1.1 Register descriptions

99.1.1.1 Read/write access control/status (RWCS)

The Read/write access control/status register provides control and status for the RWA module as shown in [Table 2111](#).

Figure 2254. Read/Write access control/status (RWCS) register

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
R	AC	RW	SZ				MAP				PR				ATTR				0	0
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CNT														ERR	DV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2110. RWCS fields

Field	Description
31 AC	Access Control. 0 End access/ Access has completed 1 Start access
30 RW	Read/Write Select. 0 Read access 1 Write access
29:27 SZ	Word Size. 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 Reserved (default to word) 100 Reserved (default to word) 101 Reserved (default to word) 110 Reserved (default to word) 111 Reserved (default to word)
26:24 MAP	MAP Select. 000 Primary memory map 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23:22 PR	Read/Write Access Priority. 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority

Table 2110. RWCS fields (continued)

Field	Description
21:18 ATTR	Access Attributes. RWA accesses are run as privileged data non-cacheable accesses by default, and drive the p_d_hprot[5:0] bus access attributes to 6'b000011 accordingly. This field is provided to allow a portion of these default values to be modified when performing read or write accesses using the RWA as shown in Table 2112 . hprot[1] must be set to one. 0xxx Reserved 1xxx Reserved x0xx p_d_hprot[4] driven to 0 for accesses x1xx p_d_hprot[4] driven to 1 for accesses xx0x p_d_hprot[3] driven to 0 for accesses xx1x p_d_hprot[3] driven to 1 for accesses xxx0 p_d_hprot[2] driven to 0 for accesses xxx1 p_d_hprot[2] driven to 1 for accesses
17:16	Reserved for future functionality.
15:2 CNT	Access Control Count. Number of accesses of word size SZ.
1 ERR	Read/Write Access Error (see Table 2111).
0 DV	Read/Write Access Data Valid (see Table 2111).

Table 2111. Read/Write access status bit encoding

Read action	Write action	ERR	DV
Read access has not completed	Write access completed without error	0	0
Read access error has occurred	Write access error has occurred	1	0
Read access completed without error	Write access has not completed	0	1
Not allowed	Not allowed	1	1

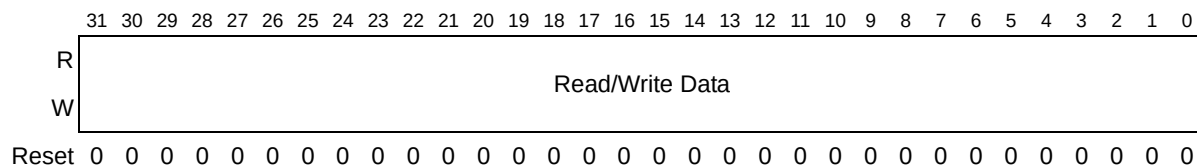
Table 2112. Protection encoding

p_hprot[4]	p_hprot[3]	p_hprot[2]	Transfer type
0	0	0	Cache-inhibited
0	0	1	Guarded, not cache-inhibited
0	1	0	Reserved
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Cachable, writethrough
1	1	1	Cachable, writeback

99.1.2 Read/write access data (RWD)

The Read/write access data register (RWD) provides the data to/from memory when initiating a read or a write access.

Figure 2255. Read/write access data (RWD) register



Read/write accesses to the memory require that the debug firmware properly retrieve/place the data in the RWD. [Table 2113](#) shows the proper placement of data into the RWD.

Table 2113. RWD data placement for transfers

Transfer size and byte offset	RWA(2:0)	RWCS[SZ]	RWD ⁽¹⁾			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	—	—	—	X
Half	x x 0	0 0 1	—	—	X	X
Word	x 0 0	0 1 0	X	X	X	X

1. X indicates byte lanes with valid data; — indicated bytes lanes with unused data.

[Table 2114](#) shows the mapping of RWD bytes to byte lanes of the read and write data buses.

Table 2114. RWD byte lane mapping

Transfer size and byte offset	RWA(2:0)	RWD ⁽¹⁾			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	XBAR[7:0]
Byte @001	0 0 1	—	—	—	XBAR[15:8]
Byte @010	0 1 0	—	—	—	XBAR[23:16]
Byte @011	0 1 1	—	—	—	XBAR[31:24]
Byte @100	1 0 0	—	—	—	XBAR[39:32]
Byte @101	1 0 1	—	—	—	XBAR[47:40]
Byte @110	1 1 0	—	—	—	XBAR[55:48]
Byte @111	1 1 1	—	—	-	XBAR[63:56]
Half @000	0 0 0	—	—	XBAR[15:8]	XBAR[7:0]
Half @010	0 1 0	—	—	XBAR[31:24]	XBAR[23:16]
Half @100	1 0 0	—	—	XBAR[47:40]	XBAR[39:32]
Half @110	1 1 0	—	—	XBAR[63:56]	XBAR[55:48]

Table 2114. RWD byte lane mapping (continued)

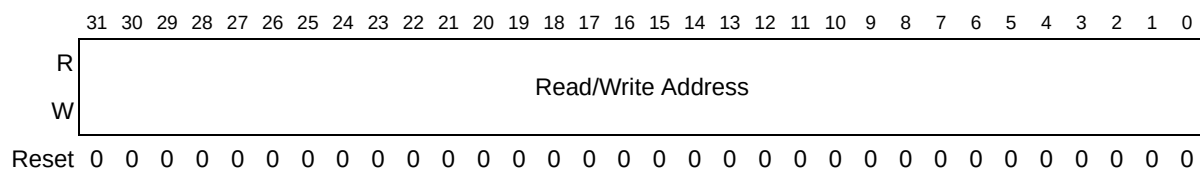
Transfer size and byte offset	RWA(2:0)	RWD ⁽¹⁾			
		31:24	23:16	15:8	7:0
Word @000	0 0 0	XBAR[31:24]	XBAR[23:16]	XBAR[15:8]	XBAR[7:0]
Word @100	1 0 0	XBAR[63:56]	XBAR[55:48]	XBAR[47:40]	XBAR[39:32]

1. — indicates byte lanes that contain unused data

99.1.3 Read/write access address (RWA)

The Read/write access address register provides the address to be accessed when initiating a read or a write access.

Figure 2256. Read/write access address (RWA) register



99.2 Nexus register access via JTAG

After the RWA module has been enabled, access to the RWA registers is enabled by loading an instruction (CORE_ENABLE = 0x7C) into the JTAG Instruction Register (IR). Once the CORE_ENABLE instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in [Table 2109](#).

Reading or writing of a RWA module register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the RWA module register to be accessed by providing an index and the direction (read/write) as shown in [Table 2109](#). This is achieved by loading an 8-bit value into the JTAG data register (DR).

This register has the following format:

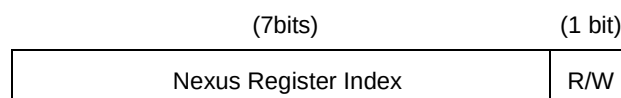


Figure 2257. JTAG DR for Nexus register access

Table 2115. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in Table 2109
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
 - a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
 - b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

99.3 Memory access examples

The following subsections describe the steps which are required to access memory.

99.3.1 Single write access

1. Initialize the RWA register through the JTAG. Configure as follows:
 - Write Address = 32h'xxxxxxx (write address)
2. Initialize the RWCS register through the JTAG. Configure the bits as follows:
 - Access Control (AC) = 1b'1 (to indicate start access)
 - Map Select (MAP) = 3b'000 (primary memory map)
 - Access Priority (PR) = 2b'00 (lowest priority)
 - Read/Write (RW) = 1b'1 (write access)
 - Word Size (SZ) = 3b'0xx (32-bit, 16-bit, 8-bit)
 - Access Count (CNT) = 14h'0000 or 14h'0001 (single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 performs a single access.

3. Initialize the RWD register through the JTAG. Configure as follows:
 - Write Data = 32h'xxxxxxx (write data)
4. The RWA module then transfers the data value from the data buffer RWD register to the memory mapped address in the RWA register. When the access has completed without error (ERR = 1'b0), the RWA clears the DV bit in the RWCS register. This indicates that the device is ready for the next access.

99.3.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in [Section 99.3.1, Single write access](#) to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The RWA module then transfers the first data value from the RWD register to the memory mapped address in the RWA register. When the transfer has completed without error (ERR=1'b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The RWA module then indicates it is ready for the next access.
3. Repeat Step 3 in [Section 99.3.1, Single write access](#) until the internal CNT value is zero (0). When this occurs, the DV bit within the RWCS is cleared to indicate the end of the block write access.

Note: The actual RWA register value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

99.3.3 Single read access

1. Initialize the RWA register through the JTAG. Configure as follows:
 - Read Address = 32h'xxxxxxx (read address)
2. Initialize the RWCS register through the JTAG. Configure the bits as follows:
 - Access Control (AC) = 1b'1 (to indicate start access)
 - Map Select (MAP) = 3b'000 (primary memory map)
 - Access Priority (PR) = 2b'00 (lowest priority)
 - Read/Write (RW) = 1b'0 (read access)
 - Word Size (SZ) = 3b'0xx (32-bit, 16-bit, 8-bit)
 - Access Count (CNT) = 14h'0000 or 14h'0001 (single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 performs a single access.

3. The RWA module then transfers the data to the RWD register. When the transfer is completed without error (ERR = 1'b0), the RWA module sets the DV bit in the RWCS register. This indicates that the RWA module is ready for the next access.
4. The data can then be read from the RWD register through the JTAG.

99.3.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in [Section 99.3.3, Single read access](#) to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The RWA module then transfers the data to the RWD register. When the transfer has completed without error (ERR = 1'b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The RWA module then indicates that the device is ready for the next access.
3. The data can then be read from the RWD register through JTAG.
4. Repeat Steps 3 and 4 in [Section 99.3.3, Single read access](#) until the CNT value is zero (0). When this occurs, the DV bit within the RWCS register is set to indicate the end of the block read access.

Note: The data values must be shifted out 32-bits at a time LSB first. The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

100 Buddy Device—LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed Debug

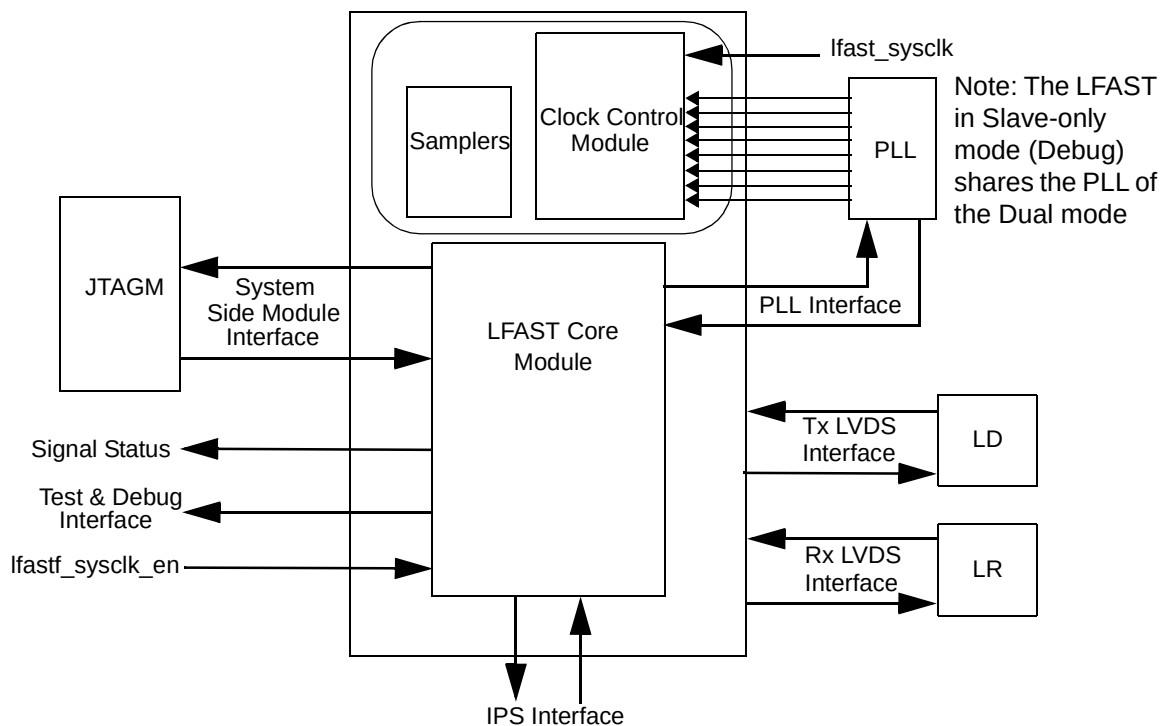
100.1 Introduction

This chapter describes the specifications of the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in slave only mode enabling high speed debug communications through the JTAG port.

100.2 Block diagram

[Figure 2258](#) depicts LFAST interaction with other modules on the device.

Figure 2258. LFAST block diagram



100.3 External signals

LFAST is a six pin interface with the following signals after enabling high speed debug mode:

- lfast_sysclk_en
 - Interface enable signal in LFAST Slave Only mode
- lfast_sysclk
 - Reference clock of the LFAST master and slave
- txdatap/txdatan
 - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
 - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface with maximum data rate of 320 Mbps.

100.3.1 LFAST operating data rates

[Table 2116](#) shows the operating modes for the LFAST interface.

Table 2116. Operating modes for LFAST interface

Interface	Tx/Rx Data Rate
LFAST Transmit	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps
LFAST Receive	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps ($\text{lfast_sysclk} \div 4$ or $\text{lfast_sysclk} \div 2$) is referred to as low and data rates 312 Mbps and 320 Mbps is referred to as high data rates.

100.4 LFAST frame structure

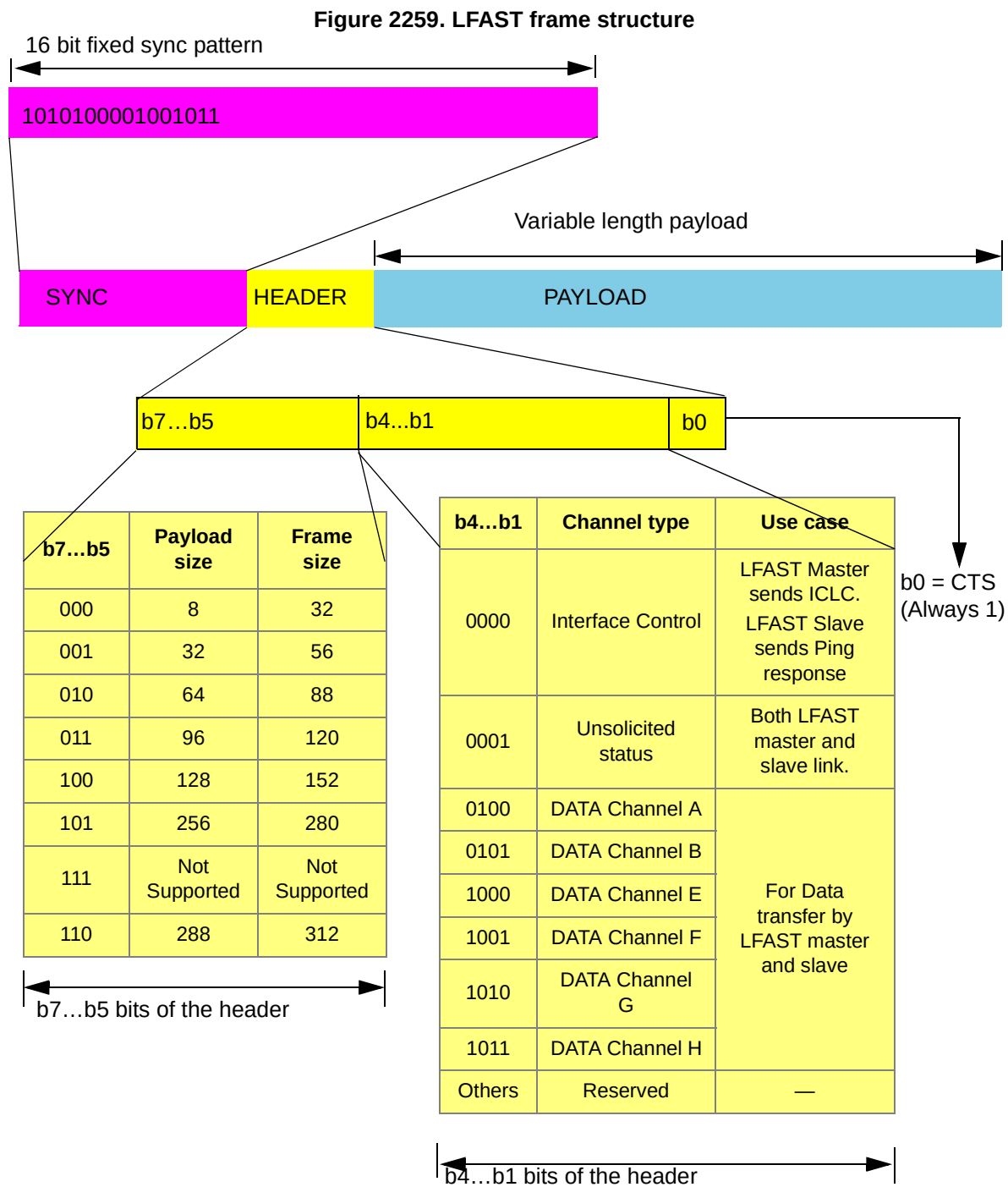
LFAST frames are transmitted and received across a serial interface using a packeted asynchronous communication format.

A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

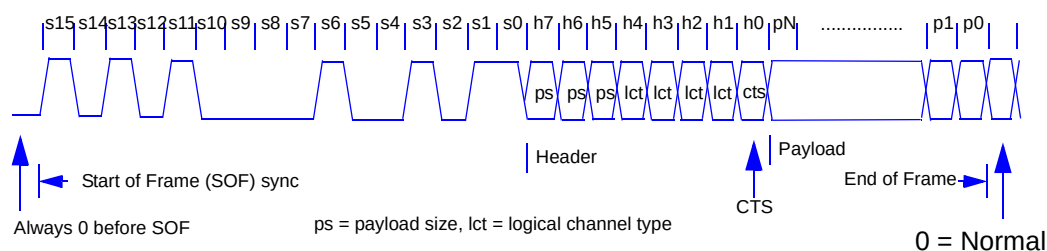
Sync pattern and header fields are of fixed length. A sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See [Figure 2259](#) for details of the LFAST frame.



The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

Figure 2260. Serial frame structure



The frame structure is shown in [Figure 2260](#) and consists of the following:

- 16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- Header - 3 MSB (b7 - b5):** Defines the payload size according to [Table](#) .

Table 2117. Header payload sizes

b7-b5 (bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	288	312
111	—	—

- Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type.
- Header - (b0):** CTS on the both LFAST master and slave devices. In slave only mode the CTS field will always be 1.
- Payload: Content dependent upon frame type.**
- Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode).

100.5 Features

- Supports slave only mode.
- Supports asynchronous data transfer rates up to a maximum of 320 Mbps
- Transmits and receives data and unsolicited frames.
- Receives ICLC frames
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports flow control using sliding window protocol.
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation.
- Supports for detection of unsupported channel number and unsupported payload size.

100.6 Memory map and register definition

All registers are 32 bits wide.

100.6.1 Memory map

Table 2118. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0000	LFAST Mode Configuration Register (MCR)	R/W	0000_0010	on page 3886
0004	LFAST Speed Control Register (SCR)	R/W	0001_0000	on page 3888
0008	LFAST Correlator Control Register (COCR)	R/W	0000_000E	on page 3889
000C	LFAST Test Mode Control Register (TMCR)	R/W	0000_0000	on page 3890
0010	LFAST Auto Loopback Control Register (ALCR)	R/W	0000_0000	on page 3891
0014	LFAST Rate Change Delay Control Register (RCDCR)	R/W	000F_0000	on page 3892
0018	LFAST Wakeup Delay Control Register (SLCR)	R/W	1201_5F02	on page 3893
001C	Reserved	—	—	—
0020	LFAST Ping Control Register (PICR)	R/W	0000_80CA	on page 3895
0024–002C	Reserved	—	—	—
0030–0038	Reserved – Access will not generate errors	—	—	—
003C	LFAST PLL Control Register (PLLCR)	R/W	0000_005C	on page 3896
0040	LFAST LVDS Control Register (LCR)	R/W	0000_5024	on page 3898

Table 2118. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0044	LFAST Unsolicited Tx Control Register (UNSTCR)	R/W	0000_0000	on page 3900
0048	LFAST Unsolicited Tx Data Registers 8 (UNSTDR8)	R/W	0000_0000	on page 3901
004C	LFAST Unsolicited Tx Data Registers 7 (UNSTDR7)	R/W	0000_0000	on page 3901
0050	LFAST Unsolicited Tx Data Registers 6 (UNSTDR6)	R/W	0000_0000	on page 3901
0054	LFAST Unsolicited Tx Data Registers 5 (UNSTDR5)	R/W	0000_0000	on page 3901
0058	LFAST Unsolicited Tx Data Registers 4 (UNSTDR4)	R/W	0000_0000	on page 3901
005C	LFAST Unsolicited Tx Data Registers 3 (UNSTDR3)	R/W	0000_0000	on page 3901
0060	LFAST Unsolicited Tx Data Registers 2 (UNSTDR2)	R/W	0000_0000	on page 3901
0064	LFAST Unsolicited Tx Data Registers 1 (UNSTDR1)	R/W	0000_0000	on page 3901
0068	LFAST Unsolicited Tx Data Registers 0 (UNSTDR0)	R/W	0000_0000	on page 3901
006C–007C	Reserved	—	—	—
0080	LFAST Global Status Register (GSR)	R	8000_0000	on page 3901
0084–0090	Reserved	—	—	—
0094	LFAST Data Frame Status Register (DFSR)	R	0000_0000	on page 3903
0098	LFAST Tx Interrupt Status Register (TISR)	R/W	0000_0000	on page 3904
009C	LFAST Rx Interrupt Status Register (RISR)	R/W	0000_0000	on page 3905
00A0	LFAST Rx ICLC Interrupt Status Register (RIISR)	R/W	0000_0000	on page 3906
00A4	LFAST PLL and LVDS Status Register (PLLLSR)	R	0002_0003	on page 3908
00A8	LFAST Unsolicited Rx Status Register (UNSRSR)	R/W	0000_0000	on page 3909
00AC	LFAST Unsolicited Rx Data Register 8 (UNSRDR8)	R	0000_0000	on page 3910
00B0	LFAST Unsolicited Rx Data Register 7 (UNSRDR7)	R	0000_0000	on page 3910
00B4	LFAST Unsolicited Rx Data Register 6 (UNSRDR6)	R	0000_0000	on page 3910
00B8	LFAST Unsolicited Rx Data Register 5 (UNSRDR5)	R	0000_0000	on page 3910
00BC	LFAST Unsolicited Rx Data Register 4 (UNSRDR4)	R	0000_0000	on page 3910
00C0	LFAST Unsolicited Rx Data Register 3 (UNSRDR3)	R	0000_0000	on page 3910
00C4	LFAST Unsolicited Rx Data Register 2 (UNSRDR2)	R	0000_0000	on page 3910
00C8	LFAST Unsolicited Rx Data Register 1 (UNSRDR1)	R	0000_0000	on page 3910
00CC	LFAST Unsolicited Rx Data Register 0 (UNSRDR0)	R	0000_0000	on page 3910

The conventions in [Figure 2261](#) and [Table 2119](#) serve as a key for the register summary and individual register diagrams.

Figure 2261. Key to Register Fields

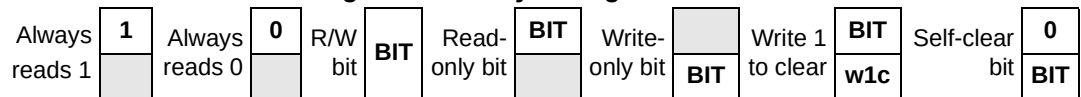


Table 2119 provides a key for register figures and tables and the register summary.

Table 2119. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bits value (other than a hardware reset).
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but its cleared by the H/W once the operation defined by the bit is over
Reset Values	
0	Resets to zero.
1	Resets to one.
[signal_name]	Reset value is determined by polarity of indicated signal.

100.6.2 Register descriptions

100.6.2.1 Mode Configuration Register (MCR)

Figure 2262. Mode Configuration Register (MCR)

Offset: 0000h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	IPGDBG	0	0	0	0	0	0	0	LSSEL
W																
Reset	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0	0	0	0	0	0	0	0	TXARBD	0	0	DRFRST ⁽¹⁾	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Set by user software and cleared by system hardware.

Table 2120. MCR field descriptions

Bits	Description
0-6	Reserved
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8-14	Reserved
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode. Refer to Section 100.10.2, Slow speed clock for more detailed description. 0 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.

Table 2120. MCR field descriptions (continued)

Bits	Description
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS LR. 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface is not disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command frame. 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 Not allowed. Do not set when in slave only mode.
19-26	Reserved
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28-29	Reserved
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after reset. 0 No soft reset. 1 Soft reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

100.6.2.2 Speed Control Register (SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Figure 2263. Speed Control Register (SCR)

Offset: 0004h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DRMD ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RDR	0	0	0	0	0	0	0	TDR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 0.

Table 2121. SCR field descriptions

Bits	Description
0-14	Reserved
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. This bit should not be modified by S/W and left as SCR[DRMD] = 1.
16 - 22	Reserved
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. This bit should not be modified by S/W.
24-30	Reserved
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. This bit should not be modified by S/W.

100.6.2.3 Correlator Control Register (COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Figure 2264. Correlator Control Register (COCR)

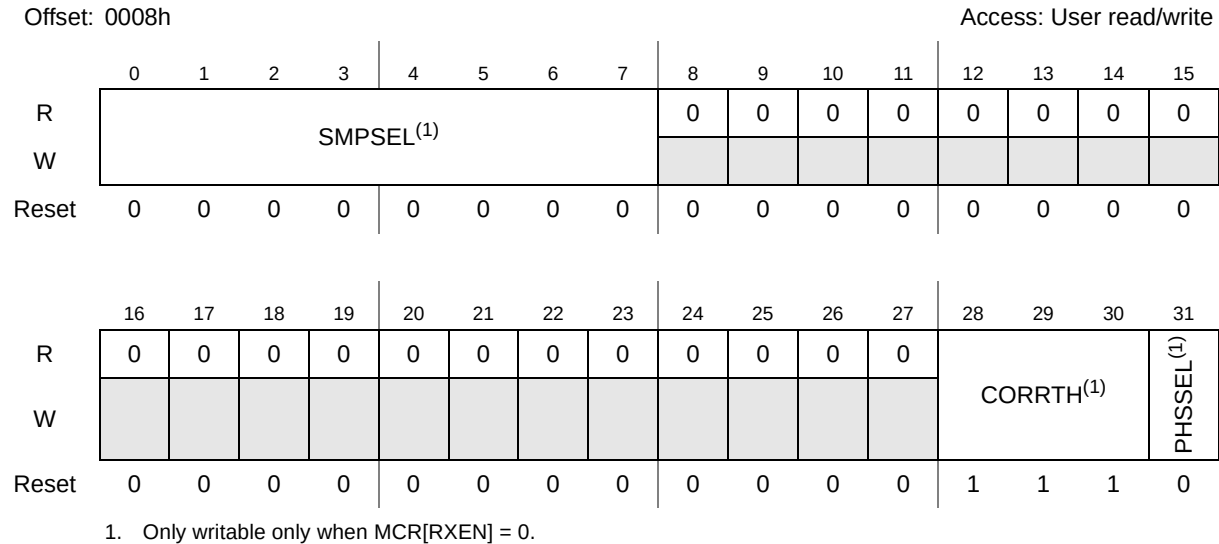


Table 2122. COCR field descriptions

Field	Description
0-7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. 00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8-27	Reserved

Table 2122. COCR field descriptions (continued)

Field	Description
28-30 CORRTH	Correlator threshold level. Defines the correlation threshold level. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. 0 8 phases 1 4 phases

100.6.2.4 Test Mode Control Register (TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes

Figure 2265. Test Mode Control Register (TMCR)

Offset: 000Ch								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	CLKTST	LPON	0	0	0	0	0	LPMOD ⁽¹⁾		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFRMTH ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Table 2123. TMCR field descriptions

Bits	Description
0-5	Reserved
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by H/W on reception of an ICLC command. 1 Clock Test mode enabled 0 Clock Test mode disabled

Table 2123. TMCR field descriptions (continued)

Bits	Description
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1 Loopback mode is enabled 0 Loopback mode is disabled
8-12	Reserved
13-15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation Others - Reserved (should not be written by S/W)
16-31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPPFDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

100.6.2.5 Auto Loopback Control Register (ALCR)

Figure 2266. Auto Loopback Control Register (ALCR)

Offset: 0010h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LPCNTEN ⁽¹⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Table 2124. ALCR field descriptions

Bits	Description
0-14	Reserved
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto predefined loopback frame transmission. 0 Infinite predefined loopback frame transmission enabled 1 Fixed count of predefined loopback frame transmission enabled
16-31 LPMFCNT	Auto Loopback Frame Transmission Count. Defines the number of predefined auto loopback frames to be sent. The predefined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload ... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

100.6.2.6 Rate Change Delay Control Register (RCDCCR)

Figure 2267. Rate Change Delay Control Register (RCDCCR)

Offset: 0014h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DRCNT ⁽¹⁾			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 1.

Table 2125. RCDCR field description

Bits	Description
0-11	Reserved
12-15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. 0h 0 cycles 1h 1 cycles ... Eh 14 cycles Fh 15 cycles
16-31	Reserved

100.6.2.7 Wakeup Delay Control Register (SLCR)

Figure 2268. Wakeup Delay Control Register (SLCR)

Offset: 0018h								Access: User read/write								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HSCNT ⁽¹⁾								0	0	0	0	LSCNT ⁽¹⁾			
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HWKCNT ⁽¹⁾								0	0	0	0	LWKCNT ⁽¹⁾			
W																
Reset	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

1. Only writable when MCR[DRFEN] = 0.

Table 2126. SLCR field descriptions

Bits	Description
0-7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. 00h 0 cycle 01h 1 cycle ... 12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8-11	Reserved
0-15	Reserved
12-15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16-23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. 00h0 cycles 01h1 cycle ... 5Fh95 cycles (1.18 us) ... FFh255 cycles Maximum
24-27	Reserved
28-31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. 0h0 cycle 1h1 cycle 2h2 cycles (1.18 us) ... Eh14 cycles Fh15 cycles

100.6.2.8 Ping Control Register (PICR)

Figure 2269. Ping Control Register (PICR)

Offset: 0020h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PNGREQ ⁽²⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PNGAUTO ⁽¹⁾	0	0	0	0	0	0	0	PNGPYLD ⁽¹⁾							
W																
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

1. Only writable when MCR[DRFEN] = 0.

2. Set by user software and cleared by system hardware.

Table 2127. PICR field descriptions

Bits	Description
0-14	Reserved
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master This bit should not be modified by S/W.
17-23	Reserved
24-31 PNGPYLD	Defines the LFAST slaves ping reply frame payload content.

100.6.2.9 PLL Control Register (PLLCR)

Figure 2270. PLL Control Register (PLLCR)

Offset: 003Ch																Access: User read/write															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	IPTMOD			0	0	0	0	0	0	0	0	0	0	0	SWPOFF	SWPON															
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	REFINV	LPCFG		0	0	PLCKCW		FDIVEN	FBDIV								PREDIV			
W																				
Reset	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0				

1. Only writable when MCR[DRFEN] = 0.

Table 2128. PLLCR field descriptions

Bits	Description
0-2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 010 Force Vctrl 011 Charge Pump Up 1 100 Charge Pump Up 2 (Internal Resistor) 101 Charge Pump Idle 110 Charge Pump Down 111 Not Used
3-13	Reserved
14 SWPOFF	SW signal to turn OFF the PLL. This bit should not be modified by S/W.
15 SWPON	SW signal to turn ON the PLL. This bit should not be modified by S/W.
16 REFINV	Invert PFD reference clock to avoid noise to other PLLs. 0 Not inverted 1 Invert

Table 2128. PLLCR field descriptions (continued)

Bits	Description
17-18 LPCFG	PLL Loop Optimization for sysclk frequency 00 1x multiplier for charge pump current 01 0.5x multiplier for charge pump current 10 1.5x multiplier for charge pump current 11 2x multiplier for charge pump current
19-20	Reserved
21-22 PLCKCW	PLL Lock Ready Count Width. Defines the number of cycles PLL waits for before asserting lock_ready flag High 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider. Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24-29 FBDIV	Feedback Division factor for VCO output clock. 00h No clock output 01h Divide by 2 (1.5, if FDIVEN = 1) 02h Divide by 3 (2.5, if FDIVEN = 1) ... 17h Divide by 24 (23.5, if FDIVEN = 1) 18h Divide by 25 (24.5, if FDIVEN = 1) ... 3Fh Divide by 64 (63.5, if FDIVEN=1)
30-31 PREDIV	Division factor for PLL Reference Clock input. 00b Direct clock passed 01b Divide by 2 10b Divide by 3 11b Divide by 4

100.6.2.10 LVDS Control Register (LCR)

Figure 2271. LVDS Control Register (LCR)

Offset: 0040h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SWWKLD ¹	SWSLPLD ¹	SWWKLR ¹	SWSLPLR ¹	SWOFFLD	SWONLD	SWOFFLR	SWONLR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0	0	0	0	0	LVRXOP			LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

1. Set by user software and cleared by system hardware.

Table 2129. LCR field description

Bits	Description
0-7	Reserved
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be put in sleep
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This bit can be changed only when DRFEN is high. This bit should not be modified by S/W.
13 SWONLD	SW signal to turn ON the LVDS LD. This bit can be changed only when DRFEN is high. This bit should not be modified by S/W.
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This bit can be changed only when DRFEN is high. This bit should not be modified by S/W.

Table 2129. LCR field description (continued)

Bits	Description
15 SWONLR	SW signal to turn ON the LVDS LR. This bit can be changed only when DRFEN is high. This bit should not be modified by S/W.
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable. The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21	Reserved
22-24 LVRXOP	LVRXOP[2] Termination resistance 0 Disable termination 1 Enable termination LVRXOP[1] Not used LVRXOP[0] Defines the LVDS LR Bias current reference
25-28	Reserved
29 LVTXOP	Control signal for LFAST and Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Data select. This bit is used for selecting use of clock sampled data or normal data in LVDS. 0 normal data to be used inside the LVDS 1 synchronised data to be used inside the LVDS
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

100.6.2.11 Unsolicited Tx Control Register (UNSTCR)

Figure 2272. Unsolicited Tx Control Register (UNSTCR)

Offset: 0044h

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	USNDRQ ⁽²⁾
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	UNSHDR ⁽¹⁾						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0

2. Set by user software and cleared by system hardware.

Table 2130. UNSTCR field descriptions

Bits	Description
0-14	Reserved
15 USNDRQ	Tx Unsolicited send request. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16-24	Reserved
25-31 UNSHDR	Tx Unsolicited message header.

100.6.2.12 LFAST Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)

Figure 2273. Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)

Offset: 0048h (UNSTDR8)....0068h (UNSTDR0)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UNTXD[31:16] ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNTXD[15:0] ⁽¹⁾															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0

Table 2131. UNSTDRn field descriptions

Bits	Description
0-31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

100.6.2.13 Global Status Register (GSR)

Figure 2274. Global Status Register (GSR)

Offset: 0080h

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0	0	0	0	0	0	0	0	0	0	0	0	LRMD	LDSM	DRSM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LPTXDN	LPFPDV	LP CPDV	LPCHDV	LP CSDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2132. GSR field descriptions

Bits	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module Slave only mode 1 LFAST Module in dual mode
1-12	Reserved
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active. 0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Recieve Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16-26	Reserved
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by Auto Loopback Control Register (ALCR) LPFMCNT. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

100.6.2.14 Data frame Status Register (DFSR)

Figure 2275. Data frame Status Register (DFSR)

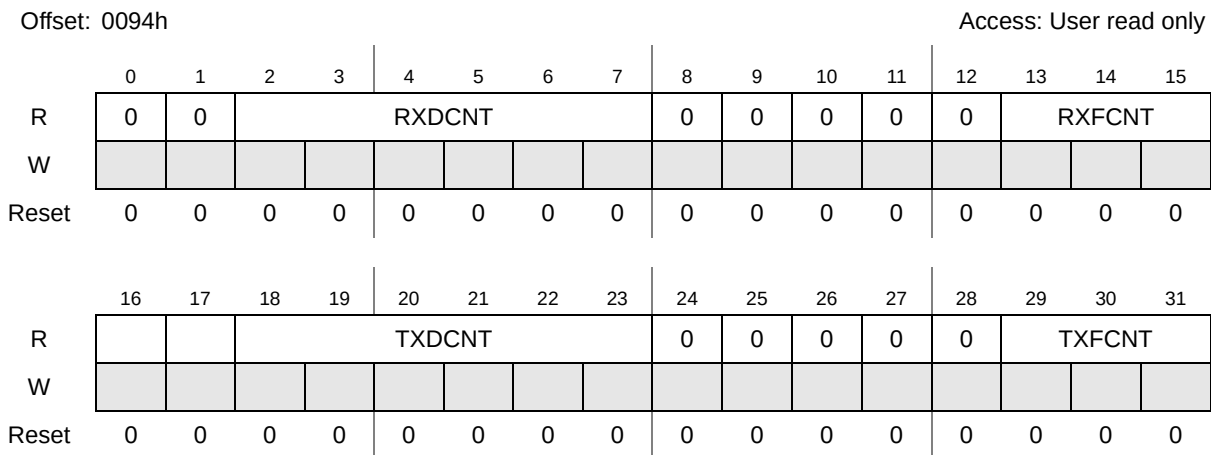


Table 2133. DFSR field descriptions

Bits	Description
0-1	Reserved
2-7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8-12	Reserved
13-15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
18-23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24-28	Reserved
29-31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

100.6.2.15 Tx Interrupt Status Register (TISR)

Figure 2276. Tx Interrupt Status Register (TISR)

Offset: 0098h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2134. TISR field descriptions

Bits	Description
0-13	Reserved
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16-26	Reserved
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28	Reserved
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

100.6.2.16 Rx Interrupt Status Register (RISR)

Figure 2277. Rx Interrupt Status Register (RISR)

Offset: 009Ch Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2135. RISR field descriptions

Bits	Description
0-7	Reserved
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. 0 Interrupt event has not occurred 1 Interrupt event has occurred- On reception of frame with payload size for a frame other than mentioned in Table 2148 .

Table 2135. RISR field descriptions (continued)

Bits	Description
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in Table 2148 .
16-27	Reserved
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
31	Reserved

100.6.2.17 Rx ICLC Interrupt Status Register (RIISR)

Figure 2278. Rx ICLC Interrupt Status Register (RIISR)

Offset: 00A0h												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2136. RIISR field descriptions

Bits	Description
0-17	Reserved
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table 2136. RIISR field descriptions (continued)

Bits	Description
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

100.6.2.18 PLL and LVDS Status Register (PLLLSR)

Figure 2279. PLL and LVDS Status Register (PLLLSR)

Offset: 00A4h								Access: User read								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PLLDIS	PLDCR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LRSPLS	LDSLPS	LDPDS	LRPDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 2137. PLLLSR field descriptions

Bits	Description
0-13	Reserved
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles. 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0
16-27	Reserved
28 LRSPLS	This bit indicates the real time status of the LR sleep signal. 0 LR sleep signal is negated. 1 LR sleep signal is asserted.

Table 2137. PLLLSR field descriptions (continued)

Bits	Description
29 LDSLPS	This bit indicates the real time status of LD sleep signal. 0 LD power sleep signal is negated. 1 LD power sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LR power down signal. When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LD power down signal. When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.

100.6.2.19 Unsolicited Rx Status Register (UNSRSR)**Figure 2280. Unsolicited Rx Status Register (UNSRSR)**

Offset: 00A8h

Access: read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	URXDV	0	0	0	0	0	URPCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2138. UNSRSR field descriptions

Bits	Description
0-22	Reserved
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24-28	Reserved
29-31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited data payload present in the Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0). The bitfield coding is same as b7-b5 in Table .

100.6.2.20 Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Figure 2281. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

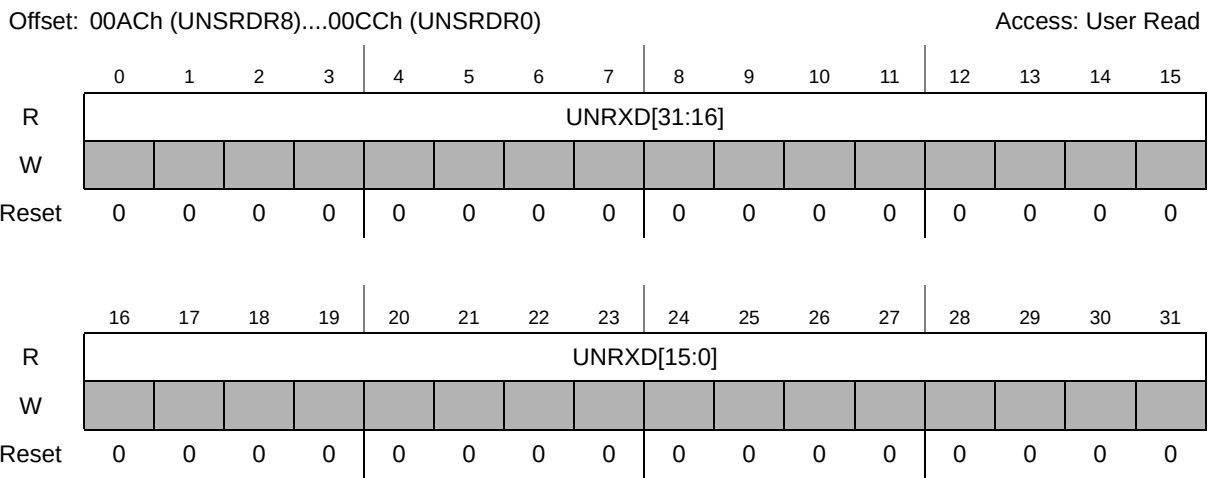


Table 2139. UNSRDRn field descriptions

Bits	Description
0-31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNSRDR0[0] in case of 288 bit payload.

100.6.3 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements^(ae). All registers of the LFAST module are classified as non-safety relevant.

The MCR[IPGDBG] fulfills the safety requirements for monitoring of the debug freeze signal.

100.7 Functional description

100.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

ae. See the “Functional Safety” chapter for register classification details.

100.7.1.1 LFAST slave interface startup procedure

1. After POR the RCDCCR are programed according to the LVDS parameter in the device datasheet.
2. The PLLCR^(af) is programed with the configuration parameters of the PLL.
3. The LCR is programed with the configuration parameters of the LVDS.
4. MCR[DATAEN] fields is programmed to select the LFAST mode.
5. Write MCR[TXARBD] = 0 to enable Tx of frames.
6. Write MCR[DRFEN] = 1 to enable LFAST.
7. Write MCR[RXEN] = 1 to enable LFAST Rx.
8. On the assertion of the LFAST interface enable the signals LR power down and LR disable are negated.
9. When an ICLC frame with payload 31h is received H/W writes RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
10. The ping frame is sent when an ICLC frame with payload 00h frame is received and H/W writes RIISR[ICPRF] = 1.

Speed mode change:

11. The PLL starts when an ICLC frame with payload 02h is received and H/W writes RIISR[ICPONF] = 1.
12. The speed of the Tx interface is changed when an ICLC frame with payload 80h is received and H/W writes RIISR[ICTFF] = 1. This will cause SRC[TDR] = 1.
13. The speed of the Rx Interface is changed when an ICLC frame with payload 10h is received and H/W writes RIISR[ICRFF] = 1. This will cause SCR[RDR] = 1.
14. The ping frame is sent when an ICLC frame with payload 00h is received and H/W writes RIISR[ICPRF] = 1.

100.7.2 LFAST Interface enable signal (lfast_sysclk_en)

The lfast_sysclk_en signal comes from the DCI module. It is asserted by the DCI internal state machine when switching from JTAG to LFAST and vice-versa can safely be done. The detailed effects of lfast_sysclk_en on the module are:

LFAST interface enable (lfast_sysclk_en) negation:

1. If LCR[SWONLD] = 0 the LD LVDS pad is powered down.
2. If LCR[SWONLR] = 0 the LR LVDS pad is powered down.
3. The LFAST slave exits the test mode (for example, loopback mode or clock test mode). Then S/W writes MCR[CLKTST] = 0 and MCR[LPON] = 0.
4. The LFAST slave Rx and Tx interfaces are set to Low Speed mode. Then write SCR[TDR] = 0 and SCR[RDR] = 0.
5. The Tx and Rx interface controllers are disabled.
6. The status registers are cleared.

LFAST interface enable (lfast_sysclk_en) assertion:

af. The PLLCR is shared between the LFAST – Interprocessor Communications and the LFAST – High Speed Debug modules. The programming values for the PLLCR in the LFAST – Interprocessor Communications are also used by the LFAST – High Speed Debug module.

1. The LD LVDS pad is power up, if all conditions are met:
 - LCR[SwoffLD] = 0.
 - LCR[SWONLD] = 1.
 - MCR[TXEN] = 1.
 - When frame ICR[ICLCPLD] = 31h is received.
2. LR LVDS pad is powered up if all conditions are met:
 - LCR[SwoffLR] = 0.
 - LCR[SWONLR] = 1.
 - MCR[RXEN] = 1.

100.7.3 Line Receiver (LR)

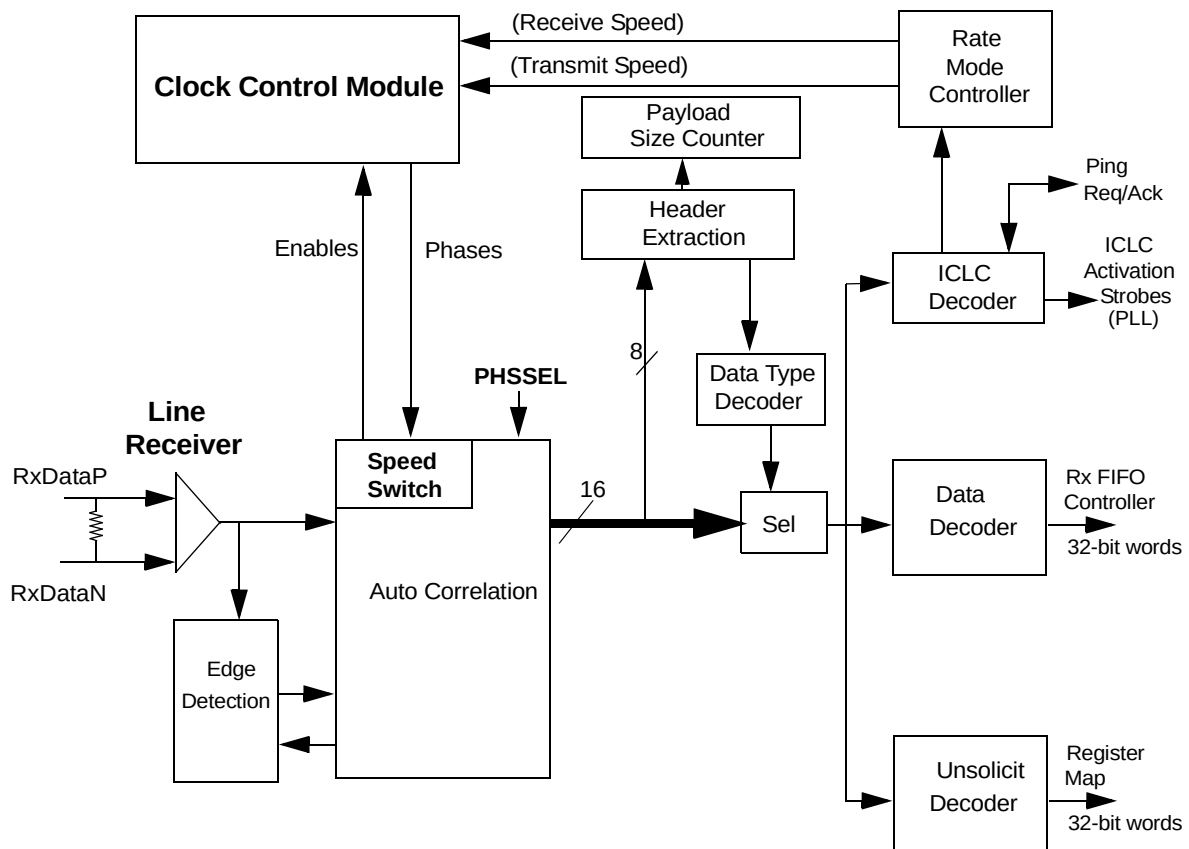
100.7.3.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 2282 shows the block diagram of Uplink Controller.

Figure 2282. Top Level Receive Controller



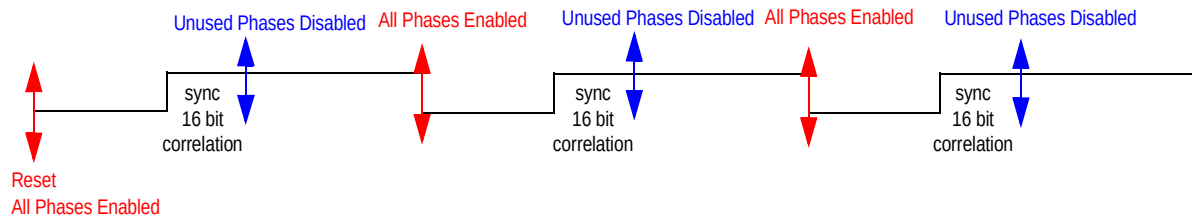
100.7.3.2 Edge-Detection & Auto correlation

The edge-detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge-detection circuitry is used or not. The edge-detection will be performed first if required before auto-correlation.

100.7.3.2.1 Auto correlation modes

100.7.3.2.1.1 Hunt correlation mode

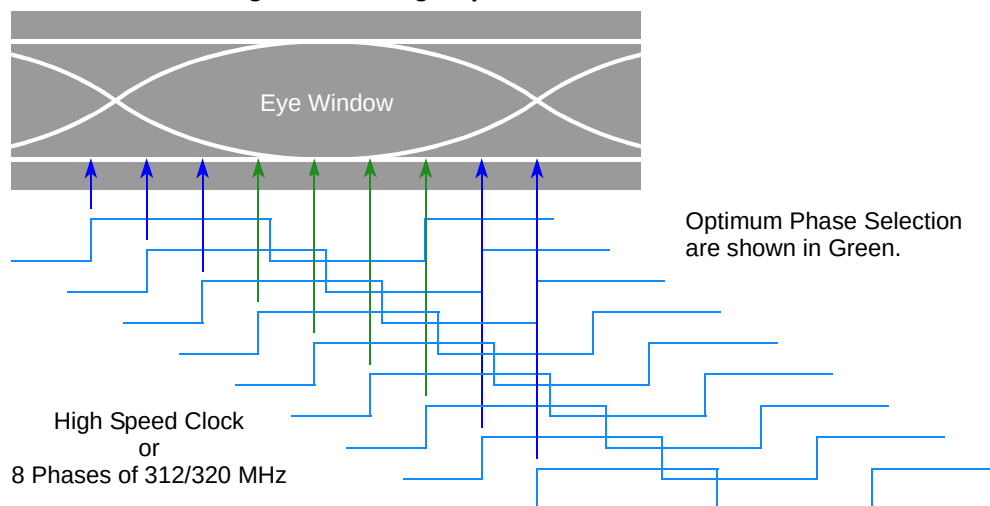
On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 2283. Hunt Correlation mode**100.7.3.2.2 Edge detection**

The edge detector is disabled in Hunt Correlation mode.

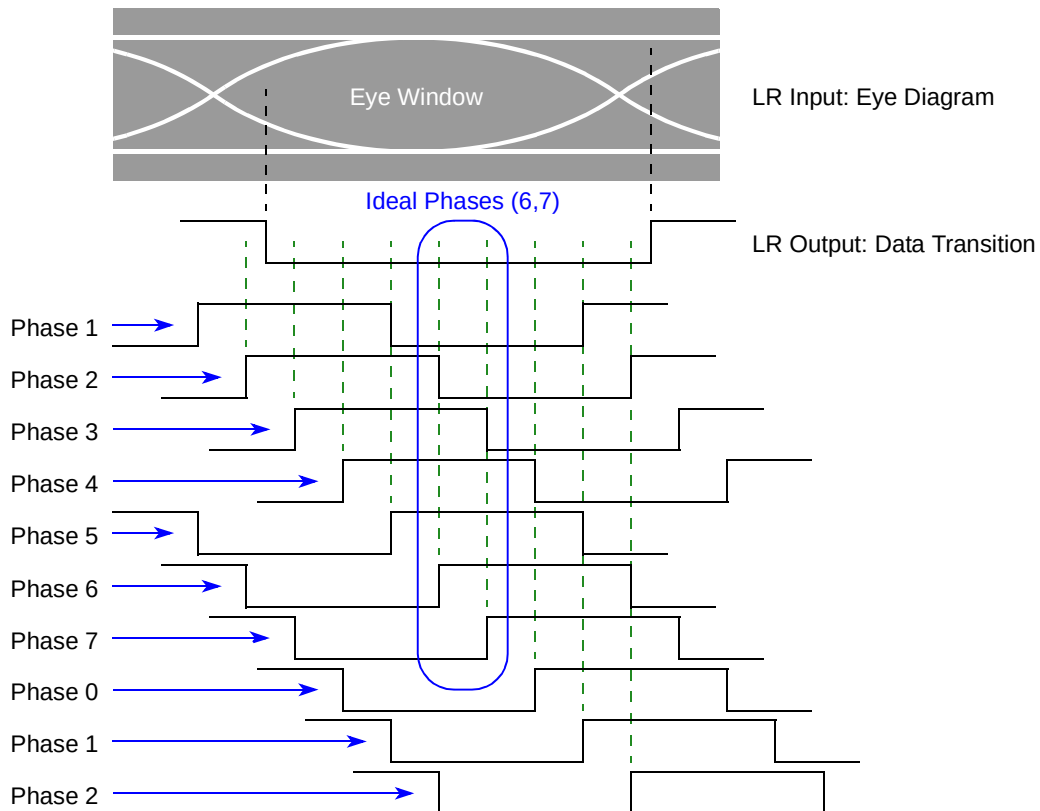
100.7.3.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in [Figure 2284](#).

Figure 2284. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in [Figure 2285](#).

Figure 2285. High Speed 8 Phase Clock Alignment Example



Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in [Figure 2286](#) and [Figure 2287](#).

Figure 2286. High Speed 4 Phase Clock Alignment Example

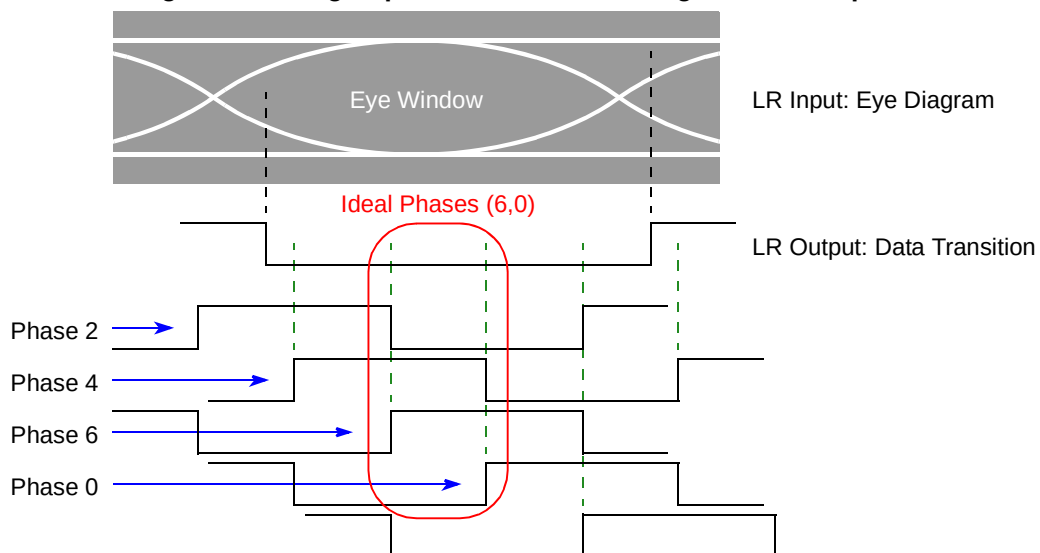
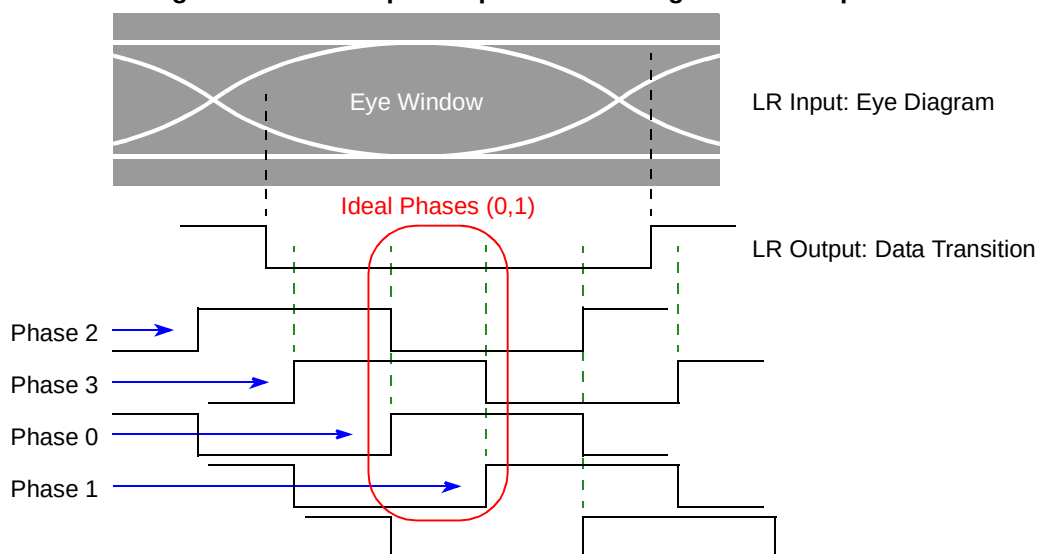


Figure 2287. Low speed 4 phase clock alignment example



The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed (312/320 MHz) 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High Speed 8 Phases: Samplers 0,1,2,3,4,5,6,7
- High Speed 4 Phases: Samplers 0,2,4,6
- Low Speed 4 Phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and re-samples it to a intermediate phase as shown in [Table 2140](#), [Table 2141](#), and [Table 2142](#) before re-sampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 2140. High speed 8 phase selection - sampling procedure

Samplers	Initial Sample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 2141. High speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See [Table 2141](#), Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 2142. Low speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See [Table 2142](#), Samplers 4, 5, 6, 7 are disabled. The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

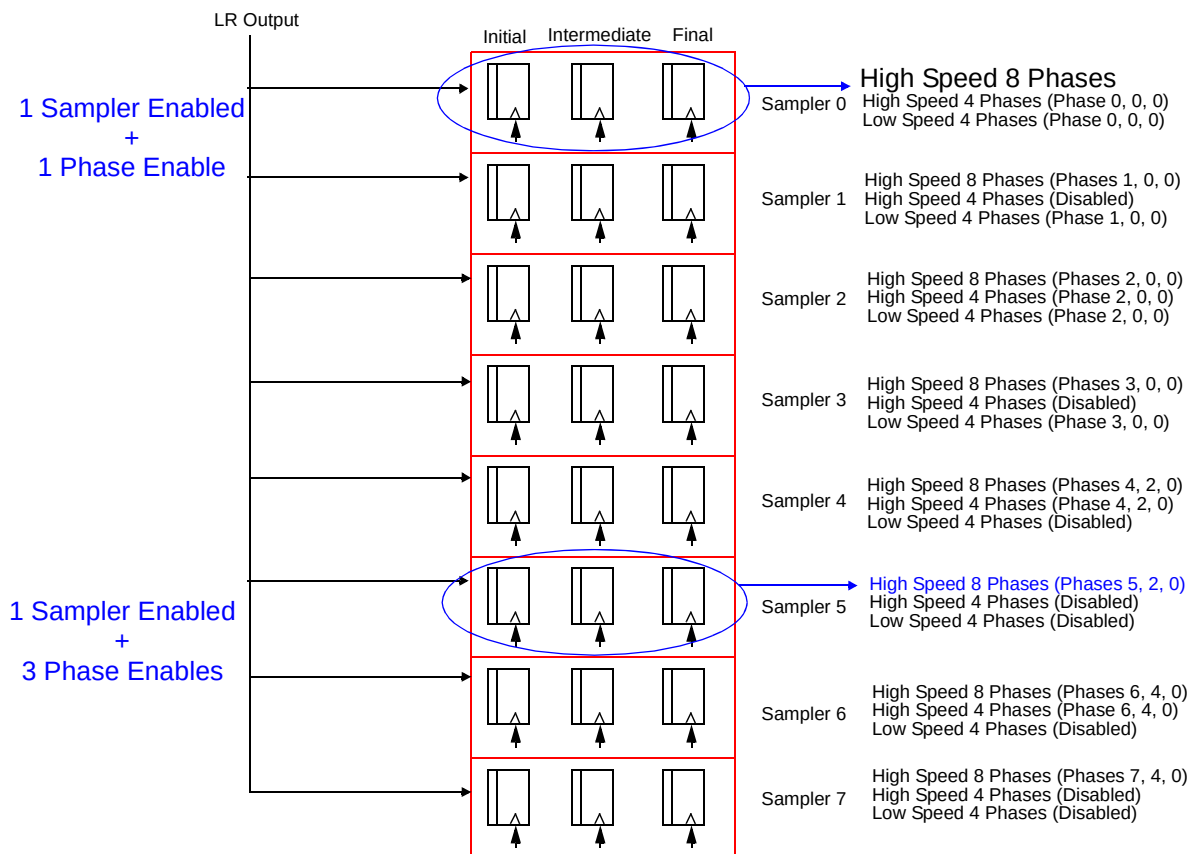
100.7.3.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in [Figure 2288](#).

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (e.g Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example, Sampler 0 has all Phase 0 content).

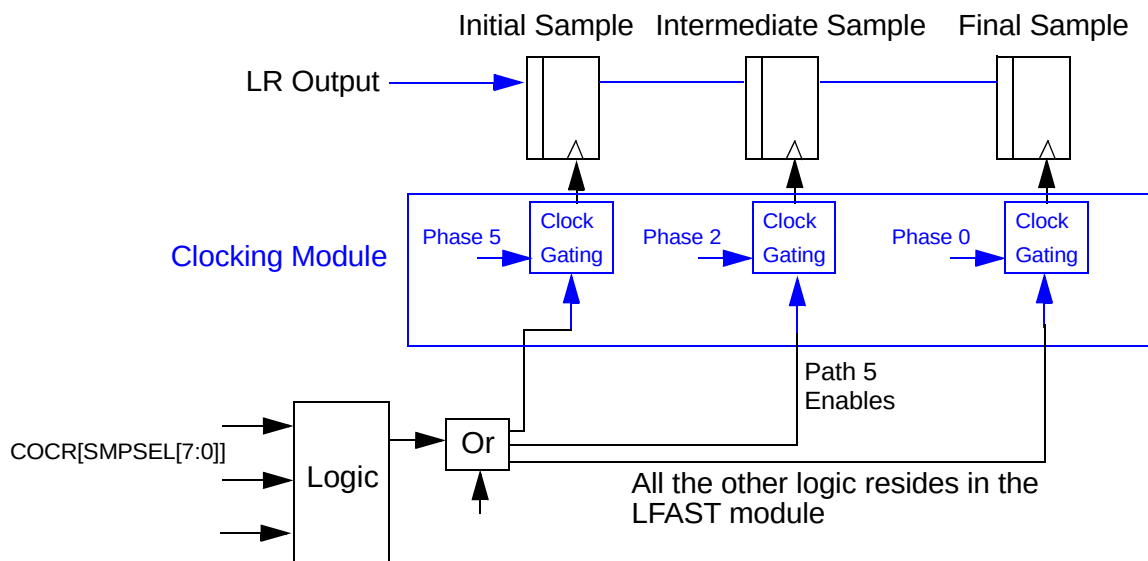
The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by de-asserting the respective phase enables.

Figure 2288. Samplers, each Sampler has 3 Registers



In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in [Figure 2289](#).

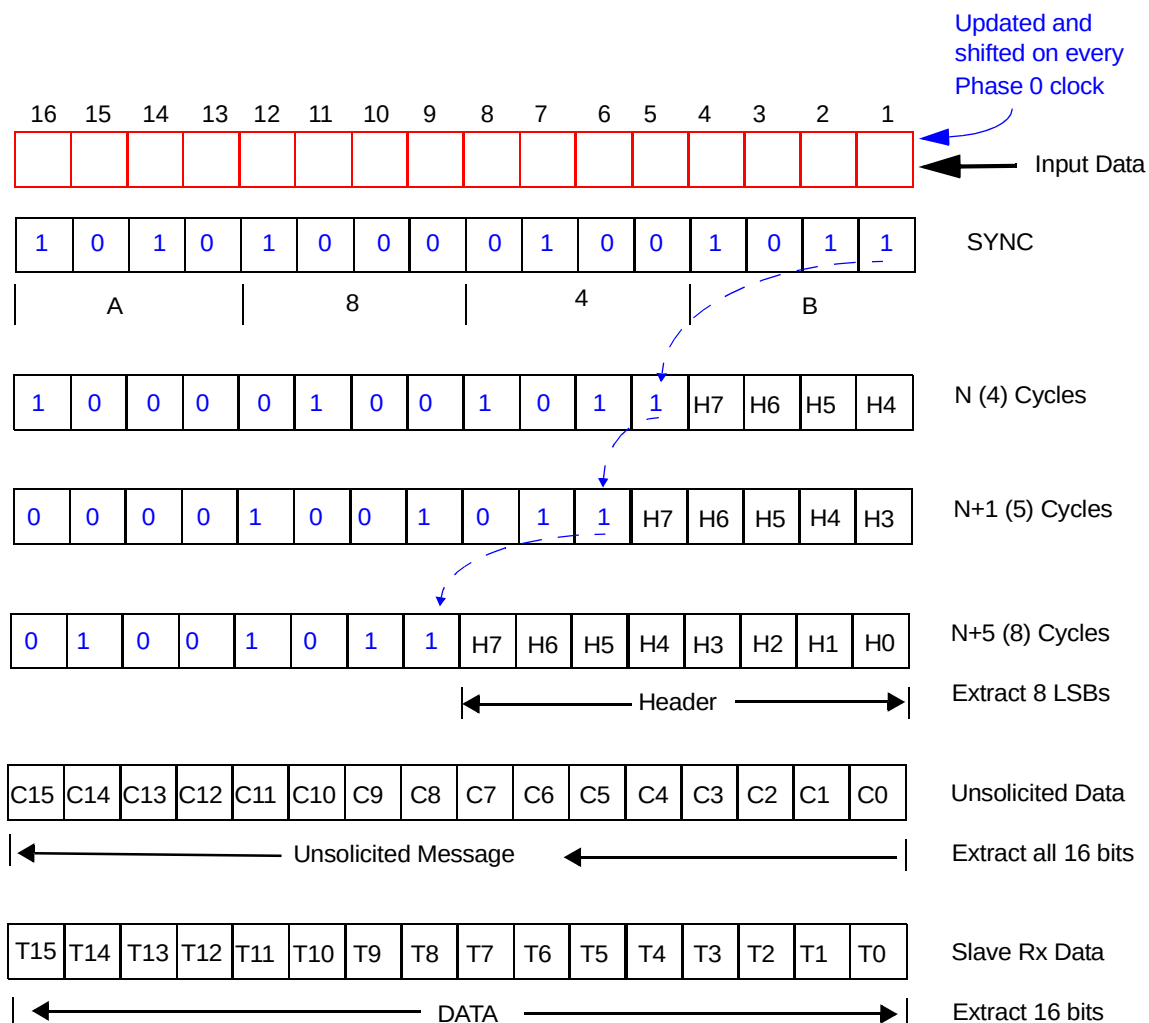
The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

Figure 2289. Sampler 5 Logic

100.7.3.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in [Figure 2290](#)

Figure 2290. Extracting Header and Payload from the 16-bit output from Auto correlation.



100.7.3.3.1 Line Receiver states

The LD has the following states:

- Shutdown**
 The LR enters in the shutdown state when the power down signal is high. In this state the LR regulator is supplying a V_{dd} level, however the LR is not enabled. In this state the LR holds its off value. Once power down signal is negated, a predefined settling time is required before the LR may be used for communication. During the settling time the peer LFAST device should not start its transfer.
- Sleep**
 The LR enters in the Sleep state when the sleep signal is high. In the sleep state, the LR is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In this

mode the receiver current consumption is reduced. Hysteresis in the LR ensures that it continues to present a 1 to the LFAST module.

- **Normal**
In the Normal state, the LR is primed for reception. Receiver converts differential signals to a single ended logic signal.

Table 2143. Line Receiver State

LR State	LFAST Slave Triggers (at least one of the following)
Shutdown	– LFAST interface enable assertion/negation – Programing of LCR[SwoffLR] or LCR[SWONLR] – Programing MCR[RXEN] and MCR[TXEN]
Sleep	– Programing of LCR[SWSLPLR] or LSC[SWWKLR]
Normal	Absence of above mentioned conditions

100.7.4 Transmit Controller

100.7.4.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, ping etc. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Recieve Controller and the ping response from the register block.

The Tx interface has two speed modes:

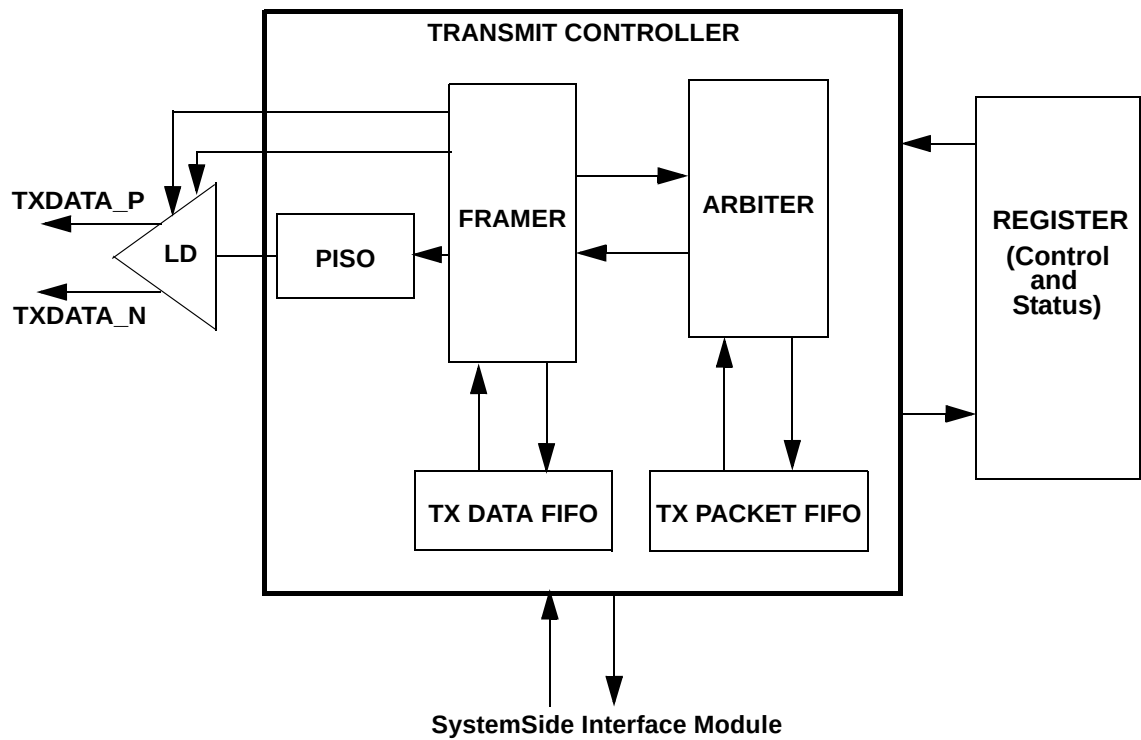
- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in [Figure 2291](#).

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

Figure 2291. Transmit Controller Connections



100.7.4.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like:

- S/W programmable registers
- System side interface FIFO
- Rx interface controller
- LFAST Slave: LFAST interface enable (lfast_sysclk_en)

The level of priority for each request is shown in [Table 2144](#).

Table 2144. Priority Levels for the Transmit Controller

Request	LFAST Slave Priority	Triggering Conditions (at least one of the listed)
LFAST interface enable (lfast_sysclk_en)	1	– LFAST interface enable (lfast_sysclk_en) is asserted – LFAST interface enable (lfast_sysclk_en) is negated
Tx Interface disabled	2	– MCR[DRFEN] = 0 – MCR[TXEN] = 0 – MCR[DRFRST] = 1 – MCR[TXARBD] = 1 – LFAST Slave: ICLC frame with payload for “Disable Rx interface” received

Table 2144. Priority Levels for the Transmit Controller

Request	LFAST Slave Priority	Triggering Conditions (at least one of the listed)
Tx Interface speed mode change	3	– SCR[TDR] is modified – LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	– TMCR[LPON] = 1 – LFAST Slave: ICLC frame with payload for loopback mode enable received <i>Note:</i> Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
Ping response request	5	– LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 – LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	– Valid only if Last Frame with header b0 = 1 received from the peer LFAST – UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	– Tx FIFO contains one or more frames – Last Frame with header b0 = 1 received from the peer LFAST device – MCR[DATAEN] = 1
Clock mode test	8	– TMCR[CLKTST] = 0 – LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCCR[DRCNT].

100.7.4.3 Line Driver digital connections

100.7.4.3.1 Line Driver states

The LD has the following states:

- **Shutdown**
The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling

time is required before the LD may be used for communication. The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In the sleep state the LD outputs are connected to V_{cm} (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a predefined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

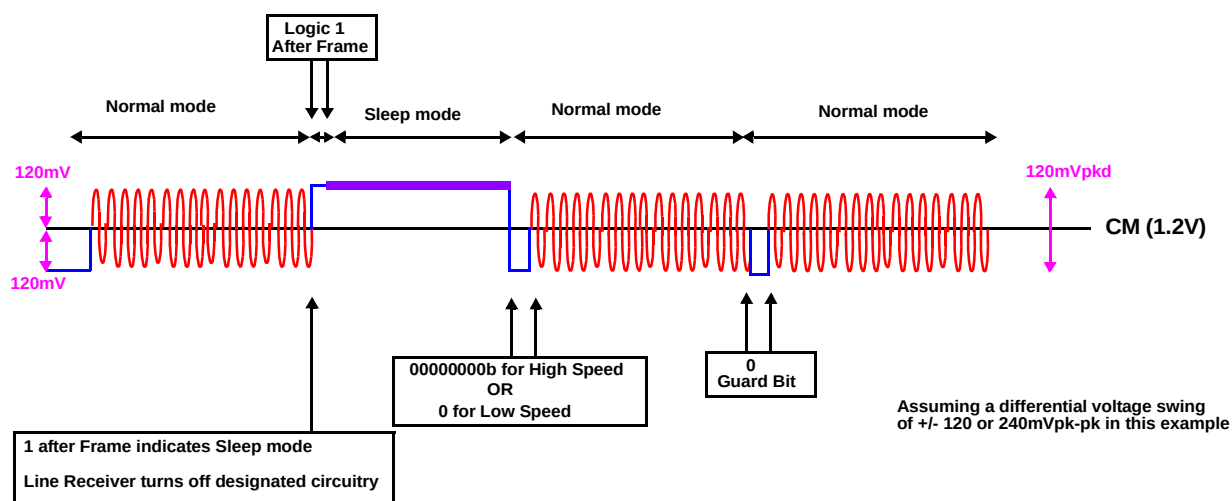


Figure 2292. Example of Sleep mode

As shown in [Figure 2292](#) before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 2145. Line Driver States

LD State	LFAST Slave Triggers
Shutdown	– LFAST interface enable (lfast_sysclk_en) negation/assertion – Programing LVDS[SWOFFLD] and LVDS[SWONLD] – ICLC command from LFAST master – Programing MCR[TXEN] and MCR[DRFEN]
Sleep	– Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions

100.7.5 Frames supported

Table 2146 provides the frames supported and their permissible payload sizes

Table 2146. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx 2. LFAST Slave Rx	Supported by 1. LFAST Slave Tx 2. LFAST Master Rx	Payload Size (bits)
Data Frame	Rx – 0100b, 1000b – 1011b Tx – All	YES	YES	TX: 96 RX: 128
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO ⁽¹⁾	8
CTS Frame	n/a	NO	NO	n/a
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

100.7.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

100.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

100.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO. Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

100.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer.

100.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[TXEN] and MCR[DRFEN] bits should be set to enable the data frame reception. When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded.

100.7.6.2 Unsolicited flow**100.7.6.2.1 Unsolicited frame transmit flow**

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

100.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
 - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
 - UNSTCR[USNDRQ] = 0
 - TISR[TXUNSF] = 1.
 - If UNSTCR[USNDRQ] = 0 then:
 - Program the unsolicited payload in UNSTD0–UNSTD8, and can be written up to a payload of frame of a maximum of 288 bits.
 - Program the unsolicited frame header in UNSTCR[UNSHDR].
 - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.

100.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTD0–UNSTD8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.

100.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is defined as a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

100.7.6.3.1 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

100.7.6.3.1.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF] = 1. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

100.7.6.3.1.2 Start PLL ICLC

The Start PLL ICLC command will write bit RIISR[ICPONF] = 1.

100.7.6.3.1.3 Stop PLL ICLC

The Stop PLL ICLC command will write RIISR[ICPOFF] = 1.

100.7.6.3.1.4 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

100.7.6.3.1.5 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write `RIISR[ICTSF] = 1` (Low speed) or `RIISR[ICTFF] = 1` (High speed).

100.7.6.3.1.6 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes `RIISR[ICTEF] = 1` and `RIISR[ICTDF] = 1`. The ICLC Tx enable command will write `MCR[TXEN] = 1`.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs `MCR[TXEN] = 1`.

100.7.6.3.1.7 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write `TMCR[CLKTST] = 1`. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates `RIISR[ICCTF] = 1` on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

100.7.6.3.1.8 Loopback payload on ICLC

This ICLC command is decoded and `TMCR[LPON] = 1` to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes `RIISR[ICLPF] = 1` on reception of this ICLC command.

100.7.6.3.1.9 Test mode off ICLC

This ICLC command is decoded and `TMCR[LPON]` and `TMCR[CLKTST]` are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable (`lfast_sysclk_en`).

100.7.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

100.7.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode `TMCR[LPFRMTH]` should be programmed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 2147. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

100.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. Bit b0 of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

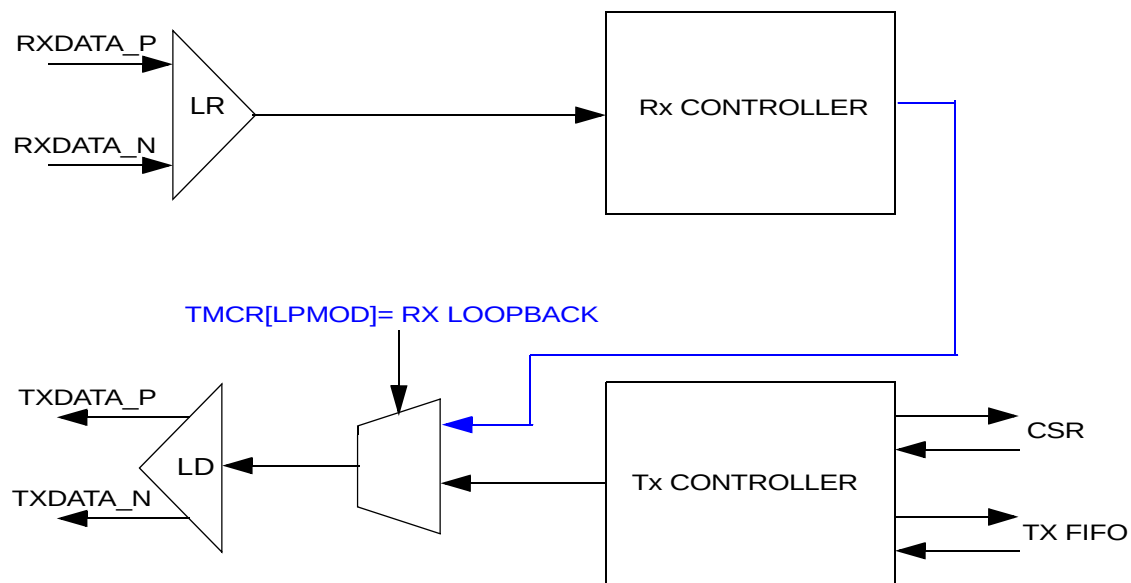


Figure 2293. Rx LoopBack mode

Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master
- For LFAST Slave: - Deassertion of the LFAST interface enable (lfast_sysclk_en).

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

100.7.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via “Rx LVDS LoopBack Mux”. Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

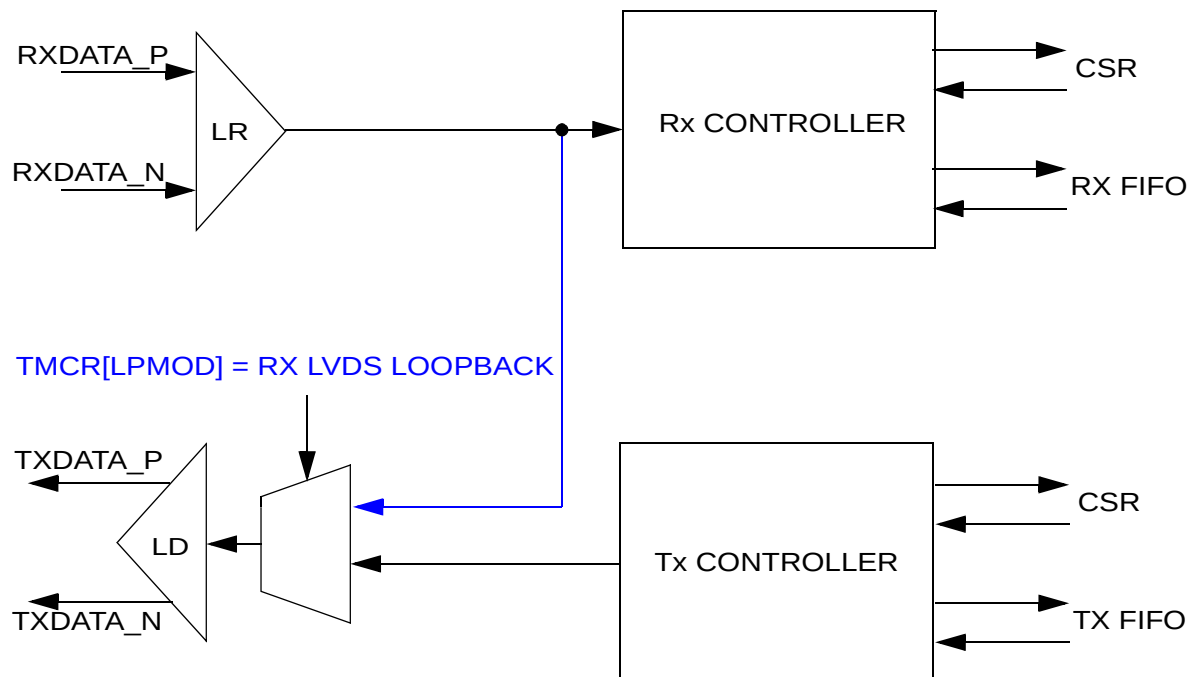


Figure 2294. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:
 - S/W writes TMCR[LPON] = 1.
 - For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
 - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.
 - Deassertion of the LFAST interface enable (lfast_sysclk_en).

100.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

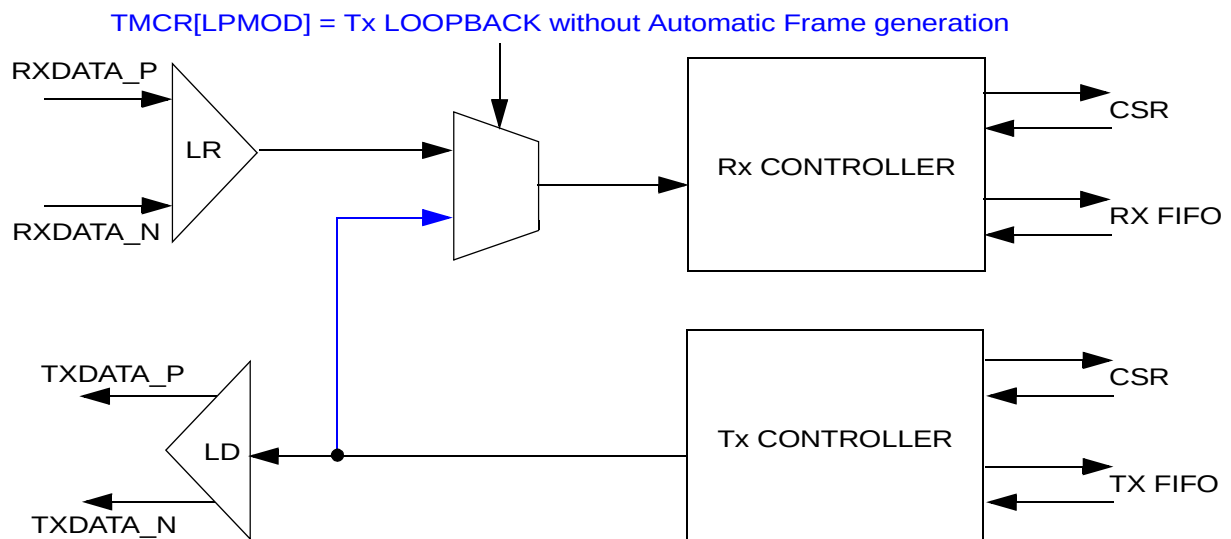


Figure 2295. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

All frames and error flags are decoded in this mode.

100.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where predefined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted.
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

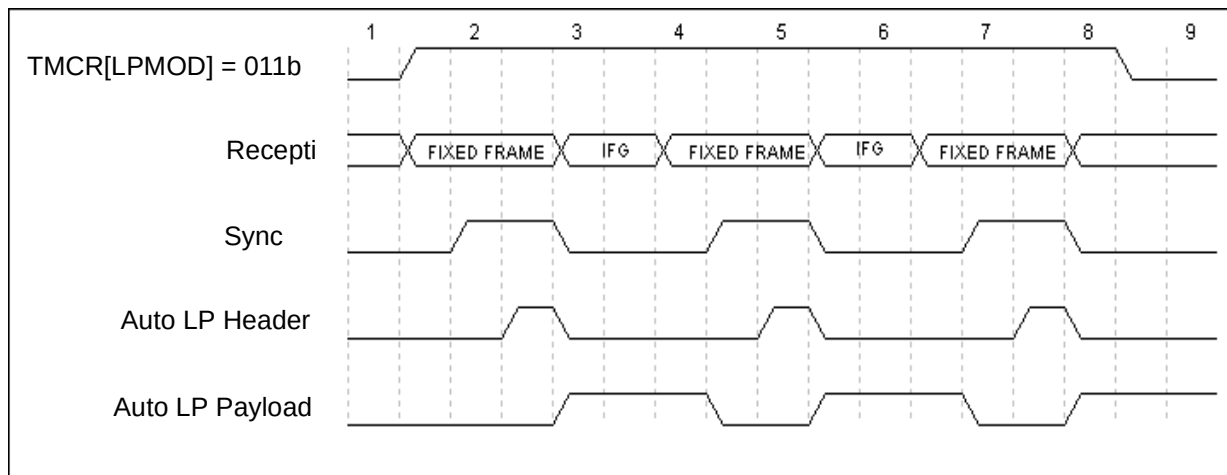


Figure 2296. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

- Tx Loopback with Automatic frame generation.
- Tx LVDS (external) with Automatic frame generation.

100.7.7.1.4.1 Tx LoopBack mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

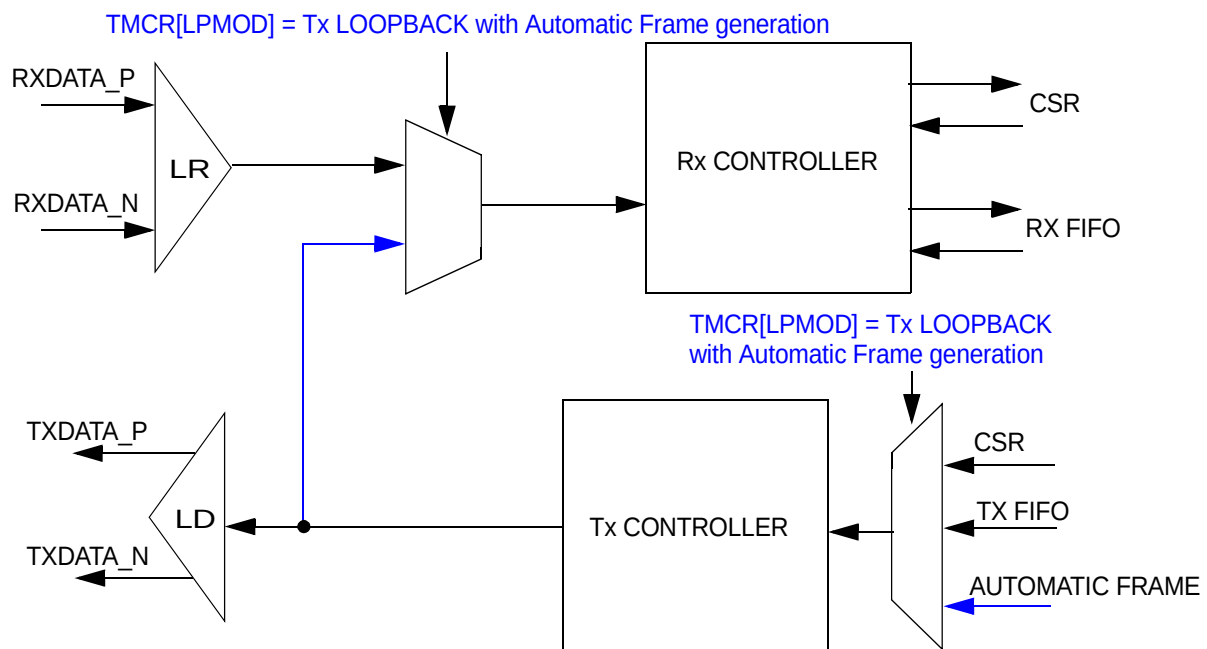


Figure 2297. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from the LFAST Master.

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

100.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

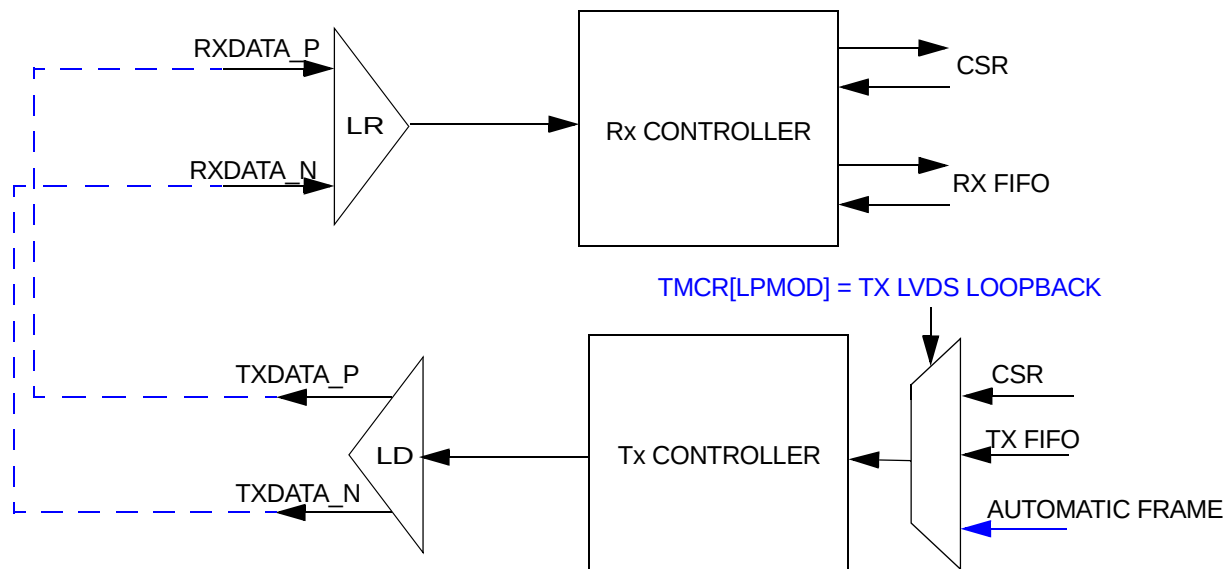


Figure 2298. Tx LVDS Loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on) from LFAST Master

Exit from Tx LVDS Loopback with automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

100.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 34h (Clock Test mode on) from LFAST Master.

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by any of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode does not affect the functionality of the Rx Interface.

100.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has its own dedicated buffer, and they are not shared between each other.

Table 2148. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer (in bits)	Rx Buffer (in bits)	Memory Type
Data Frame	6 × 32 max 2 packets	8 × 32 max 2 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	N/A	1 × 8 max 1 packet ⁽¹⁾	Registers PICR[PNGPYLD]
CTS Frame	N/A	N/A	N/A

1. Ping response packet

100.9 Resets

The various blocks in LFAST are reset as described in [Table 2149](#).

Table 2149. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space

Table 2149. Recommended receive exception handling mechanism

Reset	Blocks Reset
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ]
LFAST interface enable (lfast_sysclk_en)	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers except the following: TISR, RISR, RIISR, GSR[LPTXDN] and GSR[LPFPDV] These bits remain reset while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: Negedge only – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ] These bits does not remain reset while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: positive edge only – TISR, – RISR, – RIISR, – GSR[LPTXDN] and GSR[LPFPDV] These bits does not remain reset while the LFAST interface enable (lfast_sysclk_en) is High.

100.10 Clocks

The LFAST mainly works on three clocks:

- System Bus Clock, used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

100.10.1 Clocking strategy

[Figure 2299](#) shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

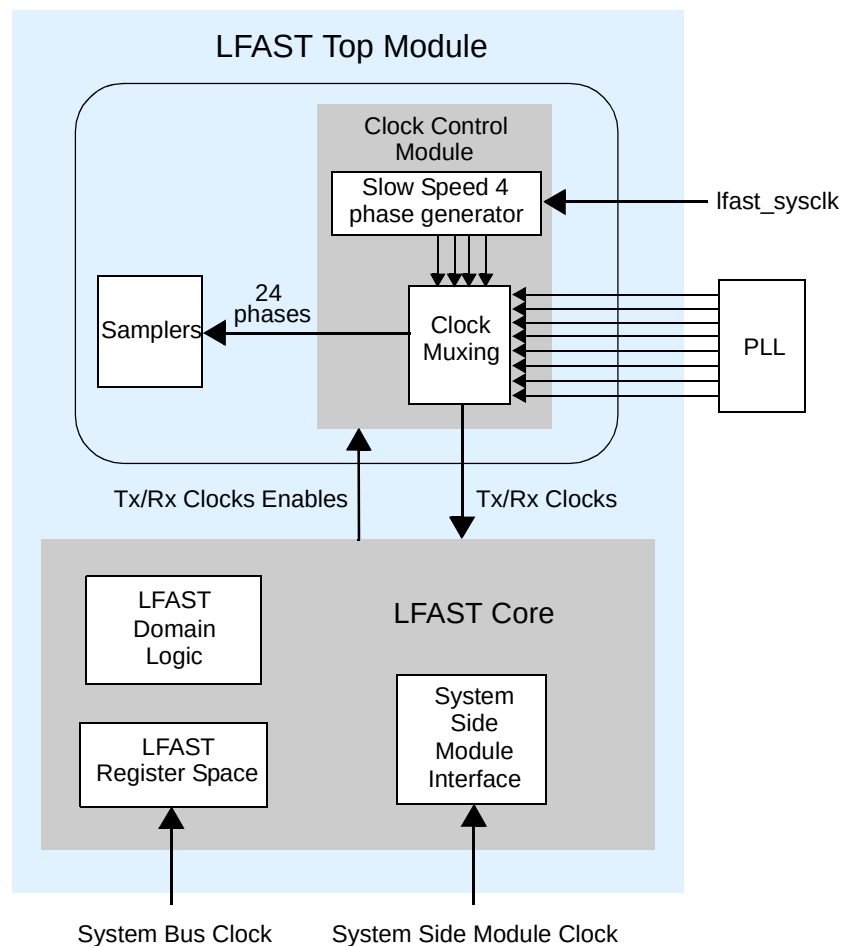


Figure 2299. LFAST module clock domains

100.10.2 Slow speed clock

100.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in [Figure 2300](#), [Figure 2301](#) and [Figure 2302](#).

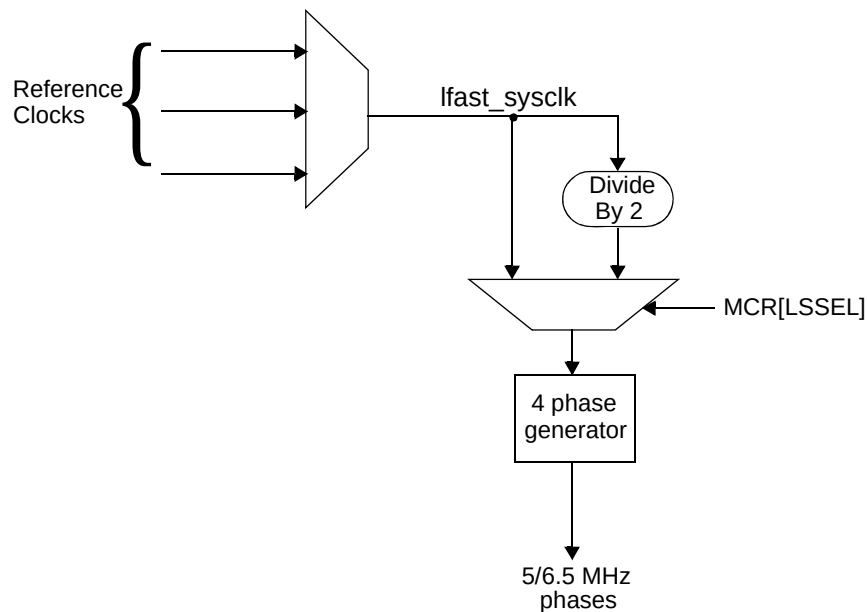


Figure 2300. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in [Figure 2300](#).

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in [Figure 2301](#) (selected clock path shown in green).

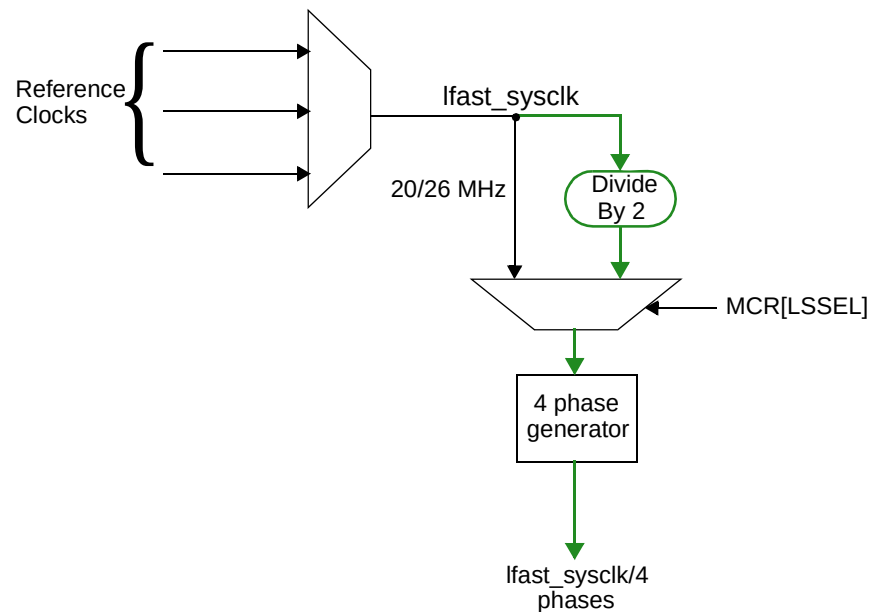


Figure 2301. External clock muxing of `lfast_sysclk` of 20/26 MHz

In this case, `lfast_sysclk` is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency `lfast_sysclk/2` when `MCR[LSSEL] = 0`. If `lfast_sysclk` is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in [Figure 2302](#) (selected clock path selected shown in magenta).

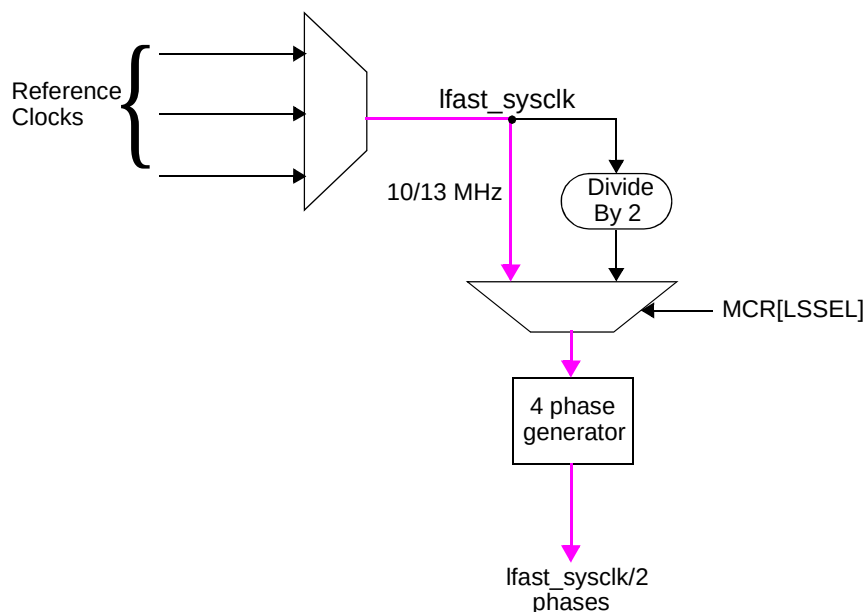


Figure 2302. External clock muxing of `lfast_sysclk` of 10 MHz

100.10.2.2 Slow speed 4-phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated.

[Figure 2303](#) Shows 4 phases getting generated using lfast_sysclk.

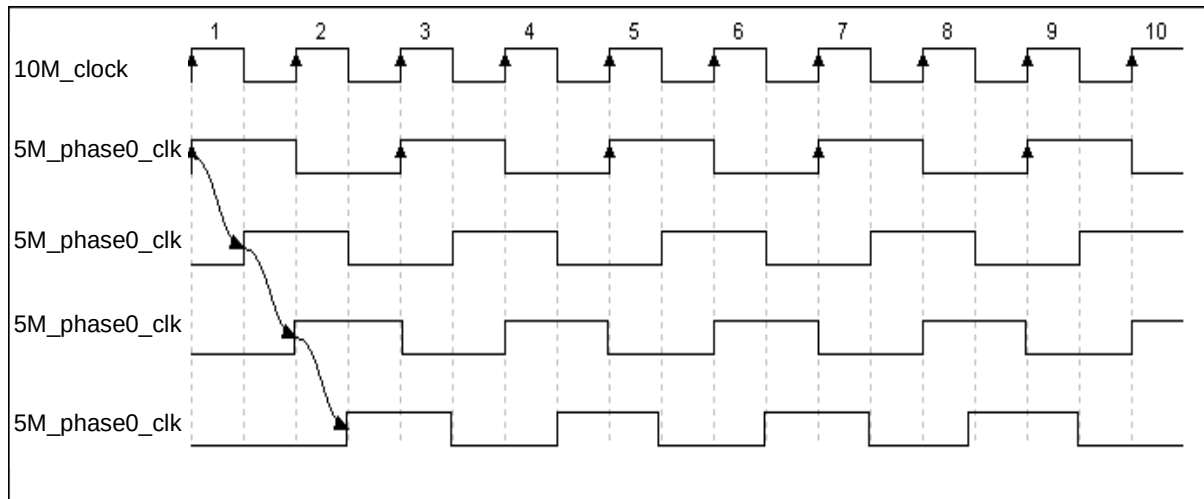


Figure 2303. Slow speed 4 phases generation

100.10.3 Rx controller clocks

100.10.3.1 Clocking module requirements for high speed phases

The high speed Phases are obtained from the PLL via the Clocking Module as shown in [Figure 2304](#).

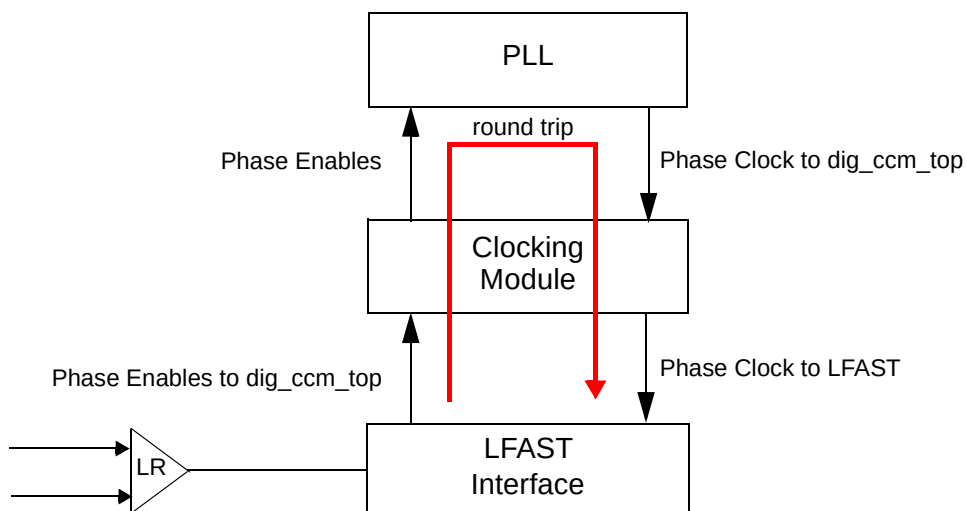


Figure 2304. Clock Enables and Clock Paths

The LFAST block will know which of the phases will be required for the different modes of operations. Therefore, the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

100.10.3.2 Clock module requirements for low speed phases

100.10.3.2.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq $\times$ 8 MHz clock from the PLL is not needed.

Table 2150. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

100.10.4 Tx controller clocks

100.10.4.1 Clocking module requirements for speed phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See [Table 2151](#).

Table 2151. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

100.10.4.2 Transmit clock muxing

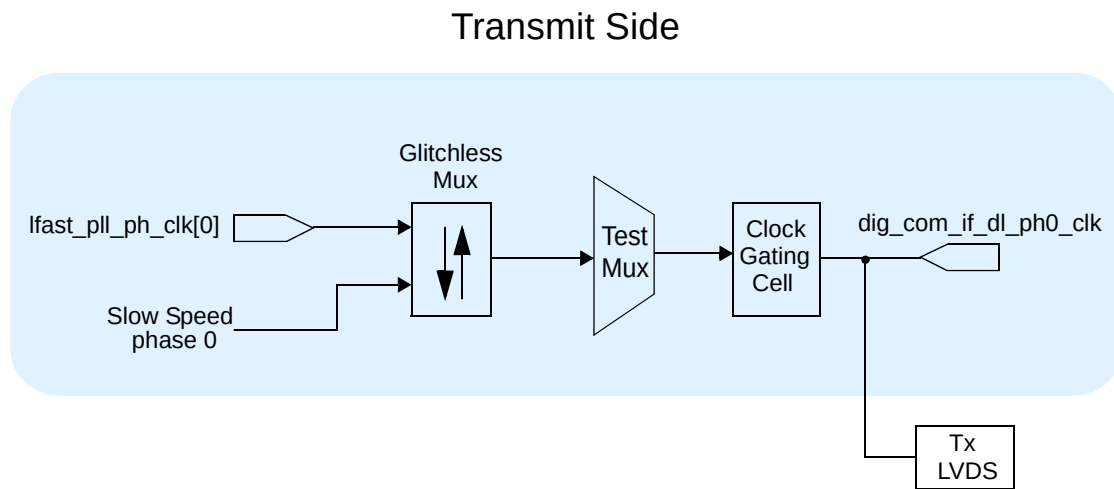


Figure 2305. Transmit Clocks Muxing

100.10.4.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

101 Revision History

Table 2152. Document revision history

Date	Revision	Changes
11-Jun-2009	1	Initial release.
15-Dec-2011	2	Throughout: Formatting, spelling, grammar, and layout corrections. Added requirement tags. Added register classification column memory map tables. Chapter Introduction Added I/O processor functional description. In the Features list table: Peripheral Core, changed Local RAM from “32 KB” to “48 KB”. Peripheral Core, changed Software Watchdog Timer from “3” to “4”. Updated SPC57EM80xx block diagram, slow and fast cross bar slave/master assignments. In SPC57EM80xx periphery allocation figure changed “3 x SWT” to “4 x SWT”. In Features list: Changed MCAN/TTCAN 3 to 3/1. Chapter Embedded Memories Changed peripheral processor core “32 KB” data SRAM to “48 KB” data SRAM. Cross-references added to memory and memory controller chapters of reference manual. Chapter Signal Description Updates made to the Power supply and reference voltage pins table. Updates made to the Pin Muxing Description Table: Added a Signals Description column. Added pin descriptions for PA1.15[0-9]. Updated DSPI pin names for Ethernet, CAN, DSPI. Updated LINFlexD and FlexRay descriptions. Updated SCK / DSPI functions. Updated SOUT / DSPI functions. Updated SIN / DSPI functions. Updated DSI / DSPI functions. Updated SAR pin descriptions. Updated “Emulation package” section description, Updated “System terminal description” table, TESTMODE pin description. Added “LVDS pins/balls” table. Updated Power supply and reference voltage pins table.

Table 2152. Document revision history (continued)

Date	Revision	Changes
15-Dec-2011	2 (continued)	Chapter Memory Map Combine adjacent reserved rows into a single row and list space in terms of MB instead of KB in some instances. Used RAM space is 128 KB. SAR_x instances renamed to SARADC_x. SD_x instances renamed to SDADC_x. Slow XBAR (XBAR_1) renamed to Peripheral Shell XBAR; Fast XBAR (XBAR_0) renamed to Computational Shell XBAR. SEMA4 changed to "Semaphore (SEMA4)". SENT_0 renamed to SRX_0. TTCAN changed to TTCAN_0. PIT_RTC_x changed to PIT_x. No RTC is implemented. DIGRF_0 is for interprocessor communication (was incorrectly stated as I/O). Terminology changed: "Mirror" flash changed to say " - overlay enabled mapping". References to "FSL" changed to "factory". BAF block removed from UTest map. Chapter Device Configuration Clarified the PRS register reset values: The computational shell's XBAR0_PRSn registers reset to 0054_3210h. The peripheral shell's XBAR1_PRSn registers reset to 0000_3210h. Added Platform Configuration Module (PCM) registers. Added FEC interface selection table. Updated DMACHMUX configuration section. Updated DMAMUX memory map table and added footnote. Updated PIT Instantiation section. Updated PIT information table to include RTI. Added PIT clocking section. Updated interrupt sources table. Updated SWT configuration section: Added additional instance, SWT3. Added note on DCI ability to change the reset value of SWT_CR[WEN]. Added "Service mode watchpoint input" sub-section. Replaced 'HSM search' in "Device specific features" with "LINFlexD search". Added "Concurrent overflow register (OFLW) implementation" section to MEMU. Updated MEMU registers table. Added "DSPI HT External Trigger" section. Added "DSPI ITSB Mode" section. Identified SENT channels in "SENT channel configuration" table.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3	Throughout Formatting, spelling, grammar, and layout corrections. Chapter Preface Added Register conventions section that details the conventions used in register figures throughout the document. Added terms to Acronyms and abbreviations table. Chapter Introduction Noted that from a software perspective, every bus master (all CPUs, DMA, SIPI, ethernet, FlexRay, HSM) sees every memory and peripheral (all RAMs, flash, and peripherals) in a consistent flat memory map Correction to features table: peripheral core has 64 KB local data RAM (was 48 KB) Detail added to features list: MCU has 4000 KB code flash, 288 KB data flash, and 16 KB UTest flash. Corrections to features list: MCU's Sigma-Delta ADC are 16-bit (previously 12-bit) There is no on-chip voltage regulator for 1.2 VChip supports both 1149.1 and 1149.7 JTAG (was 1149.1) Updates to block diagram: In Figure 1, Block diagram: Split memory block to 64KB SRAM and 64KB Standby RAM. In Figure 2, Peripheral allocation: Removed TSENS_0 and TSENS_1 and added TSENS block to PMC. Added 2 x XBIC. Chapter Embedded Memories Added list of embedded peripherals that have local memory to features list in Overview section. The core SRAM supports zero single-cycle latency on processor local accesses. Information on standby RAM and standby voltage regulator added Corrections to flash memory segmentation and read-while-write partitioning figure Added new sentence to "Processor Core Local RAM" section describing single-cycle latency. Chapter Signal Description Signal muxing table moved to SIUL2 chapter.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Memory Map - Resized reserved address range 0xF8004000 to 0xFBE03FFF (was address range 0xF8004000 to 0xFBE07FFF) - Moved SARADC1 to address range 0xFBE04000 to 0xFBE07FFF (was address range 0xFFE04000 to 0xFFE07FFF) - Moved SARADC2 to address range 0xFBE08000 to 0xFBE0BFFF (was address range 0xFBE04000 to 0xFBE0BFFF) - Changed reserved address range 0xFFE04000 to 0xFFE0FFFF (was address range 0xFFE08000 to 0xFFE3BFFF) - Moved SARADC4 to address range 0xFFE10000 to 0xFFE13FFF (was address range 0xFBE10000 to 0xFBE13FFF) - Changed reserved address range 0xFBE10000 to 0xFBE17FFF (was address range 0xFBE14000 to 0xFBE17FFF) - Moved SARADC7 to address range 0xFBE1C000 to 0xFBE1FFFF (was address range 0xFBE1C000 to 0xFBE3FFFF) - Deleted CMU_CORE (was address range 0xFBFB0200 to 0xFBFB023F) - Added Clock Monitor Unit for Fast Crossbar (CMU_FXBAR) (address range 0xFBFB0200 to 0xFBFB023F) - Deleted CMU_SYS (was at address range 0xFBFB0240 to 0xFBFB027F) - Moved CMU_SXBAR to address range 0xFBFB0240 to 0xFBFB027F (was at address range 0xFBFB0280 to 0xFBFB02BF) - Moved CMU_PRIDGE to address range 0xFBFB0280 to 0xFBFB02BF (was at address range 0xFBFB02C0 to 0xFBFB02FF) - Moved CMU_PER to address range 0xFBFB02C0 to 0xFBFB02FF (was at address range 0xFBFB0300 to 0xFBFB03FF) - Moved CMU_ADCSD to address range 0xFBFB0300 to 0xFBFB033F (was at address range 0xFBFB0340 to 0xFBFB037F) - Moved CMU_ADCSAR to address range 0xFBFB0340 to 0xFBFB037F (was at address range 0xFBFB0380 to 0xFBFB03BF) - Deleted CMU_CAN (was at address range 0xFBFB03C0 to 0xFBFB03FF) - Moved CMU_PSI5_F189 to address range 0xFBFB03C0 to 0xFBFB03FF (was at address range 0xFBFB0480 to 0xFBFB04BF) - Moved CMU_SENT to address range 0xFBFB0380 to 0xFBFB03BF (was at address range 0xFBFB0440 to 0xFBFB047F) - Deleted CMU_FLEXRAY (was at address range 0xFBFB0400 to 0xFBFB043F) - Added CMU_PSI5_F125 (address range 0xFBFB0400 to 0xFBFB043F) - Moved CMU_PSI5_1US to address range 0xFBFB0440 to 0xFBFB047F (was at address range 0xFBFB04C0 to 0xFBFB04FF) - Added reserved address range 0xFBFB0480 to 0xFBFB04FF (was cmu_psi5_f189 and cmu_psi5_1us) - Added SWT_3 (address range 0xFC05C000 to 0xFC05FFFF) - Renamed FEC at address range 0xFC0B0000 to 0xFC0B3FFF to FEC_0 - Changed reserved address range 0xFFE04000 to 0xFFE0FFFF (was address range 0xFFE08000 to 0xFFE3BFFF) - Added Tamper Detection Module (TDM) to address range 0xFFFF34000 to 0xFFFF37FFF (was reserved address range) - Split DMAMUX address space (0xFFFF6C000 to 0xFFFF6FFFF) into its 6 (DMAMUX_0 - DMAMUX_5) component spaces plus reserved address space. - Changed reserved address range 0xFFFF6D400 to 0xFFFF7FFFF (was address range 0xFFFF70000 to 0xFFFF7FFFF)

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Memory Map (continued) - Deleted CMU_IOP at address range 0xFFFFB0200 to 0xFFFFB023F - Added CMU_PLL at address range 0xFFFFB0200 to 0xFFFFB023F Name of address range 0x00400080 - 0x0040009F changed from "Customer EDC after ECC Area" to "Flash ECC Error Words Area" Changes to Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map: 0xFBE20000 to 0xFBE3FFFF reserved space added - Requirements tag SCG05.061: Covers ADD06.016, ADD06.032, ADD06.047 added - Requirements tag SCG05.062: Covers ADD06.016, ADD06.019, ADD06.027, ADD06.047) Changes to RAM memory map - RAM address space split into two 64 KB ranges—64 KB standby RAM and 64 KB system RAM. 0x52800000 to 0x5280FFFF changed to 64 KB used (was 48 KB Changes to UTest flash memory map: 0x004000C0 to 0x004000C3 Soft DCFRecord Start Address record added 0x004000C4 to 0x004000FF Reserved space added - Address 0x00400000 contains a calculated constant used in die temperature calculation. - Address 0x00400002 contains a calculated constant used in die temperature calculation. Chapter Functional Safety New chapter Chapter Device Configuration Miscellaneous changes: "AIPS" changed to "PBRIDGE" New sections added: - Password and Device Security Module (PASS) configuration - Tamper Detection Module (TDM) Configuration Core modules section: - Footnote added to e200z4 reset settings table stating the levels that determine the bits are set by the factory and may change between revisions of the device. Crossbar switch configuration section: - Additional details added SMPU configuration section: - SMPU_HSM information moved to the <i>SPC57EM80xx Microcontroller Security Reference Manual</i> - New subsection added: Master Access permissions for SMPU0 - New subsection added: Master Access permissions for SMPU1

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Device Configuration (continued) Platform Configuration Module (PCM) section: • FBOMCR[ACCERR] field definition updated. • AHB_BE1[RSV] bit reset values changed to 0 (was undefined). • IAHB_BE1[RSV] field definition changed to read, "Reserved bits define the PRE, BRE and BWE of the bus gasket for the HSM." • IAHB_BE2[RSV] bit reset values changed to 0 (was undefined). Interrupt Controller (INTC) interrupt sources section: • Major rewrite of entire section. • Revised the descriptions and names for IRQs 558–593. • Deleted entries for IRQs 264, 265, 267, 273, 274, 276, 282, 283, 285, 291, 292, 294, 372, 373, and 375. • IRQ 266: removed DSPI_0_SR[DPEF]. • IRQ 275: removed DSPI_1_SR[DPEF]. • IRQ 284: removed DSPI_2_SR[DPEF]. • IRQ 293: removed DSPI_3_SR[DPEF]. • IRQ 374: removed DSPI_12_SR[DPEF]. New subsection added: Interrupt Latency DMACHMUX section updates: • DMAMUX peripheral DMA request to input source mapping table updated • Block diagram updated Memories and memory interfaces section: • New subsection added: Flash memory controller (PFLASH) configuration Timers section: • Secure SWT function clarified • New subsection added: Secure watchdog DSPI section: • Updated DSPI instantiation table, no LVDS support for DSPI2. • Updated DSPI2 instantiation for PCS signals. • Updated DSPI2 Instantiation for LVDS support and indicated DSPI12 availability in half duplex mode. CAN subsystem configuration section: • Updated MCAN CREL field values as per latest MCAN integration. DigRF and Signal Interprocessor Interface: • Block diagram updated New section added: Indirect Memory Access (IMA) configuration FCCU configuration section: • New subsection added: FOSU parameters • Register names changed to align with FCCU chapter Added Security Modules section, which includes: • PASS module • Tamper Detection Module (TDM)

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Device Configuration (continued) Updated SIPI/DigRF diagram and content Register protection section: Added SIUL2 protected registers: SIUL_IFER0, SIUL_IFMC0-SIULIFMC31, SIUL_IFCPR Offsets of all PMC_DIG protected registers have changed. Number of PMC_DIG protected registers in groups REE_VDn, RES_VDn, and CTL_TDn has changed. Added new section: IMA (Indirect Memory Access) protected registers Added new MC_RGM protected register group: RGM_PRSTn Chapter Reset and Boot No significant changes. Chapter Device Configuration Format (DCF) Records Added figure after UTEST DCF client table to describe bit positions of the UTEST Miscellaneous Register. Chapter Power Management Updated "Low power support and STOP mode implementation" section Deleted voltage regular safety implementation figure Flash uses the externally supplied 1.25 V for the low voltage supply, while high voltage (3.3 V) is being generated by a dedicated internal regulator. Chapter Security Noted that advanced security features require significant involvement Chapter Calibration and Debug Added text to core debug support section Added text to calibration section Added text to Debug DigRF section Added Debug over CAN section Chapter e200z4d Core Complex Overview EFPU2 Floating-Point Unit features section added LSP Lightweight Signal Processing Unit features section added Added "sign/zero bits" to bulleted item in Section 13.3.5: LSP Lightweight Signal Processing Unit features: Specialized operations such as field manipulation, count-leading sign/zero bits Chapter Core (z420n3) Overview In table for Machine Check Syndrome Register (MCSR) field descriptions, corrected positions of reserved bits (from 10:11 to 9:10) Vector integer non-MAC and Vector integer MAC columns were removed from Concurrent instruction issue capabilities table Vector integer MAC row was removed from Concurrent instruction issue capabilities table Footnote was added to Machine Check Syndrome Register regarding the setting of the MEA bit. The name of the DCECE bit was changed to DCPECE for all occurrences. The bit description of bit 21 of the DMEMCTL0 register was modified to include the stbdcbx, sthdcbx and stwdcbx instructions. The bit description of bit 22 of the DMEMCTL0 register was updated for clarity. The name of the ICECE bit was changed to ICPECE for all occurrences.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Core (z425n3) Overview In table for Machine Check Syndrome Register (MCSR) field descriptions, corrected positions of reserved bits (from 10:11 to 9:10) The Cache Registers L1CSR1 SPR 1011, L1FINV0 SPR 1016 and L1FINV1 SPR 959 were changed to L1CSR0 SPR 1010, L1CSR1 SPR 1011, and L1FINV1 SPR 959. Updated Concurrent instruction issue capabilities table to reflect additional capabilities. A footnote was added to the Machine Check Syndrome Register to reflect that the z420n3 does not have address translation so the MEA bit is always set to 0. Chapter System Integration Unit Lite2 (SIUL2) Signal muxing table added—was formerly in Signal Description chapter. Chapter Crossbar Switch (XBAR) Added sentence in Functional Description that describes how the crossbar returns all slave response information back to the requesting master. Chapter Crossbar Integrity Checker (XBIC) No significant changes Chapter Peripheral Bridge No significant changes Chapter System Memory Protection Unit (SMPU) No significant changes Chapter Intelligent AHB Gasket (IAHBG) No significant changes Chapter Semaphores2 (SEMA42) No significant changes Chapter Interrupt Controller (INTC) No significant changes Chapter Enhanced Direct Memory Access (eDMA) In Section 24.5.7, Dynamic programming removed subsection, "Dynamic channel linking and dynamic scatter/gather," and replaced with new subsections, "Dynamic channel linking" and "Dynamic scatter/gather". Chapter DMACHMUX No significant changes

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Clocking Updated device clock generation figure by adding the Ethernet clock selector and divider and modified the CAN clock divider to 1...64. Updated the "MC_CGM relationship to clocks" table to show the Ethernet (REF_CLK) being sourced by Auxiliary Clock 11 Select. Updated the "Clock generation" figure: - Removed duplicate clock source for the Ethernet (REF_CLK) that showed Aux. Clock 2 as the source - Added the Ethernet clock selector and divider and modified the CAN clock divider to 1...64. Updated all instances of μs representing microsecond to ms. Added "System Clock/GTM Output Clock/Ethernet Clock Pin Muxing" section. Added "TTCAN/MCAN clocking" section. Changed the signals names CLKOUT0, CLKOUT1 and SYSCLK to SYSCLK0, SYSCLK1 and CLKOUT, respectively. Chapter Dual PLL Digital Interface Updated equations 3 and 4 to $f_{VCO0} = \dots$ and $f_{VCO1} = \dots$ Chapter Clock Monitor Unit (CMU) Updated the CMU_FDR[FD] field description by changing CMU_MCR[MD] to CMU_MDR[MD] in the included formula. Chapter Clock Generation Module (MC_CGM) Aligned the register names in the memory map to those in the register descriptions. In the CGM_AC11_SC figure, changed the address offset (was 0x0940, is 0x0960). In the CGM_AC11_SS figure, changed the address offset (was 0x0944, is 0x0964). In the CGM_AC11_DC0 figure, changed the address offset (was 0x0928, is 0x0968). Revised the "Progressive system clock switching" section. Added the "Fractional divider details" section and a cross-reference to it from the CGM_AC5_DC0 section. Chapter OSC Digital Interface (XOSC) Reset value of XOSC[OSCBYP] updated to 1. Changed XOSC_CTL[RSTBYP] reset value to device specific entry of —. Updated XOSC_EN bit location in UTEST Misc row to 7 in "XOSC configurations" table. Chapter RCOSC Digital Interface Removed DEM_EN bit of CTL register Chapter RAM Controller (PRAM) No significant changes

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Flash Memory Controller Added new bullet to Calibration Overlay Summary describing per-master calibration remap capability. Added details to describe that calibration remapping will take effect immediately the cycle after PFCRCR[GRMEN] is enabled. PFDCDRE register renamed to PFCBRE Following deleted from Section 1.5.1.7.1, PFlash Calibration Region Descriptor n, Word0 (PFCRDn.Word0): “The hardware performs basic checks on the validity of the logical start address when this word is written; if a non-supported logical start address is defined, the register write is terminated with an error and the register left unchanged and the valid bit cleared.” Correction to PFlash Calibration Region Descriptor n, Word0 (PFCRDn.Word0) LSTARTADDR field description in Table 9 (PFlash Calibration Region Descriptor n, Word 0 description): bit range is 0-27 (was 31-4) Deleted from Section 1.5.1.7.2, PFlash Calibration Region Descriptor n, Word1 (PFCRDn.Word1): “Attempted writes to this register with illegal values, as defined by the sizes and locations of the overlay memories, are terminated with an error and clear the valid bit.” Correction to Section 1.6.19.2, Flash address generation check: failure in address generation or control logic or transmission path is forwarded to the FCCU (was MEMU). Also deleted following: “ The PFLASH_C55FM flash memory controller has a separate MEMU reporting interface associated with each AHB port, p0 and p1.” Chapter Embedded Flash Memory (c55fmc) Adjusted the definition of PFCR1[APC] Added timing diagrams for address pipelined accesses and read buffer misses Removed Interface Signal block diagram in Section 1.4 as it did not add any appreciable value to the document Updated Figure2 to depict two, separate on-chip RAM interfaces Added a warning that the programming model should not be updated while the flash controller is in use. Removed unused Section 1.7 Timing Diagrams Added new section: On-Chip Overlay RAM Feedback Check Corrected an error in the e2eECC calculation example under ‘Safety Considerations’ Added new bullet to Calibration Overlay Summary describing per-master calibration remap capability. Added details to describe that calibration remapping will take effect immediately the cycle after PFCRCR[GRMEN] is enabled. Chapter Decorated Storage Memory Controller (DSMC) No significant changes Chapter Configuring Flash Memory Write and Read Protection New chapter

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Analog-to-Digital Converters (ADC) Configuration Updated figure "Sigma-Delta ADC integration diagram" with Sigma-Delta input mux wrap-around and advance feature as well as watchdog trigger function and output to GTM channel. In "ADC overview", changed "256 FusionQuad ED" to "216 FusionQuad ED" Added "Sigma-Delta Mux Input Advance and Wrap Around" section Updated figure "SAR ADC integration diagram" to add SAR watchdog trigger output to GTM channel. In SAR ADC analog input pin multiplexing section: changed "Analog input pins in the 292 BGABGA292 that only connect to SAR ADCB use a single analog switch pad cell." to "Analog input pins that only connect to SAR ADCB use a single analog switch pad cell." Throughout: - CLKEXT changed to EXT_CLK - BEXT changed to EXT_DATA - Vddm changed to VDD_HV_ADV - Vssm changed to VSS_HV_ADV - Vss changed to VSS_LV - Vdd changed to VDD_LV - SARB changed to SARADC_B - SAR ADC or SARn -> SARADC or SARADC_n Removed "Sigma-Delta ADC external signal description" and "SAR ADC external signal description" sections as they are a duplicate of what is in the pin description chapter. Sigma-Delta ADC integration diagram: Updated signal names: MUX_ADV, DMA_IRQ_EN, VALID_DATA_AVAILABLE, WATCHDOG_LIMIT_TRIGGER In Clock Skew section, changed "decimation" to "oversampling", removed last sentence of first paragraph, removed 2nd paragraph (single sentence). Changed section heading from Clock/data inputs to "External modulator clock/data inputs" Changed section heading from Clock output to "External modulator clock outputs" Changed section heading from Conversion result interrupt/DMA gating signal to "Conversion result Interrupt/DMA" In "SAR ADC clock sources" section, changed 14.4 to 14.6: "The maximum frequency for the clock input to the converter is 14.6 MHz" In "SAR ADC diagnostic" section, changed VAREF1 to VDD_HV_ADR_S, and VAGND1 to VSS_HV_ADR_S - Updated figure: "Analog input pin pullup/pulldown pad cell" - Updated figure: "Analog input AN35 — discharge switch to AVSS" - Updated table: "SAR analog input channel assignment" - Updated table: "SARADC_B analog test channel assignment" and changed input channel 105 to VDD_HV_IO_EBI. Added the following to "Conversion result interrupt/DMA" section: "A DMA or interrupt can be generated for each conversion for each ADC. The Sigma-Delta ADC has an 8-deep result FIFO." Added "Conversion Limit Watchdog Output to GTM" section. Added "TOM/ATOM" signal names to "SARADC input trigger diagram" figure. Also added shaded box and changed "System Integration" to "SIUL2" Added following note to Section 39.2.2.2, "SAR ADC analog input pin multiplexing": "Simultaneous sampling of two ADCs on same analog input is not allowed." In Table 584 (SARADC_B analog test channel assignment), changed Description for input channel 109 from Reserved to VAGND1 (SARADC ground reference). Added CRES bit to SARx_CTR0–3 register definition tables. Added NTRGEN and NEDGESEL fields to SARx_MCR registers.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface Added four new conditional text tags • WDG_monitor_no in Table 618., Conditional Text Tags. Added two new features in Section 40.3, Features Added a new row in Table 602., SDADC Digital interface memory map Added 1-bit wide fields 'WDGEN' and 'WRMODE' in Section 40.6.2.1, Module Configuration Register (MCR). Added 3-bit wide field 'ANCHSEL_WRAP' in Section 40.6.2.2, Channel Selection Register (CSR). Added description of 'wraparound mode' in 'ANCHSEL' field of Section 40.6.2.2, Channel Selection Register (CSR) Added 3-bit wide field 'ANCHSEL_CNT', two 1-bit wide fields 'WTHH' and 'WTHL' in Section 40.6.2.4, Status Flag Register (SFR). Added two 1-bit wide fields 'WTHDIRS' and 'WTHDIRE' in Section 40.6.2.5, Request Select and Enable Register (RSER). Added a new register, Section 40.6.2.10, WDG Threshold Register (WTHHLR) Added 'Wraparound Control Mechanism' description in Section 40.7.8, Data conversion and Section 40.7.9, Hardware triggering. Added a description of 'WDG threshold crossover event' in Section 40.7.10, Interrupt/DMA request support. Chapter Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface Added a new sub-feature in Section 41.3, Feature description. Updated Section 41.4.8, Programmable analog watchdog Added new bit, WTRIGCUT in Section 41.5.1.1, Main Configuration Register (MCR). Chapter Temperature Sensor Following text added to Section 4.3.1, Temperature threshold detection (digital output generation): The threshold detection circuit is calibrated in respect to the factory-test temperatures: the low temperature threshold is trimmed to match the cold insertion test temperature (nominally equal to –40C) and the high temperature thresholds are trimmed such that one of them matches the hot insertion test temperature (nominally equal to 150 °C or 165 °C). In other words, the thresholds are not trimmed to absolute values of temperature but to the specific temperatures for which the device is validated during factory test. Chapter Timers Overview No significant changes Chapter System Timer Module (STM) No significant changes Chapter Software Watchdog Timer (SWT) Features section, 2nd buttlet. Changed to 'Internal 16MIRC oscillator clock is used for timer operation'. SWT_CR: Replaced CSL bit with Reserved. Periodic Interrupt Timer (PIT) No significant changes

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter CAN Subsystem • In the MCAN CCR register field descriptions table, corrected bit 29 to ASM and bit 30 to CCM. • In the MCAN Bit Timing and Prescaler Field Descriptions table, removed last two rows. • In the MCAN ECR figure and field descriptions, corrected bit 16 as RP and bit 17:23 as REC. • In the MCAN Global filter configuration register field descriptions table, corrected bits 28:29 as ANFE. • In the MCAN Standard ID Filter Configuration register figure, corrected LSP[7:0] to LSS[7:0] • In the MCAN Extended ID Filter Configuration register, corrected FLSSA[15:2] to FLESA[15:2] • Added a note in the M_CAN Feature list • In the MCAN Rx FIFO 1 Status register figure, added new bits 0:1 as DMS • Updated Standard Message ID Filter Element table • Updated Extended Message ID Filter Element • Added a note in the M_CAN BTP register • Updated MCAN Message RAM Configuration diagram • Added Interface to DMA Controller section • In Detailed mapping in CAN subblocks table, in the first comment, replaced SIDFC[SAFLS] with SIDFC[FLSSA]. In the second comment, replaced XIDFC[SAFLE] with XIDFC[FLESA]. Added a new row for debug messages MCAN block. • In the M_CAN message memory size requirement table, added, added a new row for the Rx Status FIFO 1 element. • Removed reference to PBRIDGE1 from the CAN subsystem generic block diagram • Updated MCAN CREL and TTCAN CREL register figure footnote. • Updated M_CAN message memory size requirement table • Added debug messages rows for each MCAN instance in the “Updated Shared memory map vs CAN transmit, receive and filters elements” table. • Updated MCAN Standard Message ID Filter Element table and figure • Updated MCAN Extended Message ID Filter Element table and figure Chapter Serial Interprocessor Interface (SIPI) No significant changes Chapter Deserial Serial Peripheral Interface (DSPI) • Renamed SDR to SDR0, ASDR to ASDR0, COMP1 to COMP0, DDR to DDR0, SSR to SSR0, DIMR to DIMR0, and DPIR to DPIR0. • Added New 32 bit Registers SDR1, ASDR1, COMP1, DDR1, SSR1, DIMR1, DPIR1, PISR4, PISR5, PISR6, PISR7, CTARE0-7, SREX. • Updated Features section about 8 extended CTAR registers; programmable serial frame size upto 64 bits; DMA Support for CMDFFF and DDIF; Interrupt Support for CMDFFF, CMDTCF, TFIWF. • Updated DSPI Memory Map • Added XSPI bit to MCR register • Added FMSZ6 to CTAR_SLAVE Register • Added BSYF, CMDTCF, CMDTFFF and TFIWF fields to SR register. • Added CMDTCF_RE, CMDTFF_DIRS, CMDTFFF_RE, TFIWF_RE, DDIF_DIRS fields to RSER Register.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Deserial Serial Peripheral Interface (DSPI) (continued) • Updated PUSHHR, TXFRn Register Descriptions. • Added FMSZ[5] field to DSICR0 • Updated "Serial Peripheral Interface (SPI) Configuration" functional description to mention about CMD FIFO and the new XSPI Mode and its usage. • Updated "Deserial Serial Interface (DSI) configuration" functional description to mention about the new DSI64 Mode and its usage. • Added a Note to "Multiple Transfer Operation (MTO)". • Updated "Combined Serial Interface (CSI) configuration" to include effects of the new XSPI and DSI64 modes. • Added "Command FIFO Fill Interrupt or DMA Request", "Transmit FIFO Invalid Write Interrupt Request", Command Transfer Complete Interrupt Request", "Deserialized Data Match DMA Request" to "Interrupts/DMA Requests". • Added "Address Calculation for first-in-entry and last-in-entry in the CMD FIFO. • Updated figures "DSPI Block Diagram", "DSI Serialization Diagram", "DSI Deserialization Diagram", "TSB Downstream Frames", and "TSB Data Frame Format for MSC Dual Receiver Operation". • Updated PUSHHR[CONT] bit description. • Updated Stop Mode, Module Disable Mode descriptions under Power Saving Feaures. • Updated DSPIx_PUSHHR and DSPIx_PUSHHR_SLAVE Register descriptions. • Added content to De-serial Serial Interface (DSI Mode) Functional Description. • Modified constraint on FMSZ field in DSPI_CTAR Register description. • Updated MCR[XSPI] register bit description. • Updated DSPI_PUSHHR description Chapter DigRF - Interprocessor Communications Added TISR, RISR and RIISR to the "DigRF memory map table. Chapter Fast Ethernet Controller (FEC) In the "FEC register map" table, changed "FIFO Receive FIFO Start Register" to "FIFO Receive Start Register" to align to the actual register name. In the "Functional description" section, added the "MII data frame" and "MII management frame structure" subsections. Added introductory content to the "Functional description" section. Chapter FlexRay Communication Controller (FlexRay) StopWatch logic description added Updated PLL Clocking section Chapter Inter-Integrated Circuit (I2C) Modified section 1.6.3.3 and 1.6.5.2 for stop mode generation in DMA Rx mode Moved content from the IBFD register description to its own section, "Clock rate and IBFD settings".

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Peripheral Sensor Interface (PSI5) Section 51.3.2.5, General Interrupt Control Register (PSI5_GICR): added field IE_INVALID_CMD_LEN Section 51.3.2.9, General Interrupt Status Register (PSI5_GISR): added field IS_INVALID_CMD_LEN Section 51.3.2.30, Data Output Block Configuration Register (PSI5_DOBCR): updated various fields Section 51.3.2.36, Data Preparation Register Low (PSI5_DPRL): updated to remove "Sensor Address and control Region" bits and to renumber the "Command Payload Region" bits Section 51.3.2.38, Data Buffer Register Low (PSI5_DBRL): updated register fields and note Section 51.3.2.39, Data Buffer Register High (PSI5_DBRH): updated register fields and note Section 51.3.2.40, Data Shift Register Low (PSI5_DSRL): updated register fields and note Section 51.3.2.41, Data Shift Register High (PSI5_DSRH): updated register fields and note Updated Section 51.4.3.4, Data transmission Updated Section 51.4.3.4.2, Data Preparation Register (64-bit register, DPRH + DPRL, DPRH is fixed at 0) Updated Section 51.4.3.4.3, Data Buffer Register (64-bit register) Updated Section 51.4.3.4.4, Data Shift Register (64-bit register) Updated Section 51.4.5, Interrupt handling Updated figure Data transmission block Updated figure ECU-to-sensor communication registers (implementation of Figure 1026) Change from MK Cut1: Updated the Stop mode description in Section 51.1.3.4, Stop mode Updated the description in Section 51.1.3.5, Debug mode to explicitly include HardWare clearing bits and the DMA message registers effected. Change from MK Cut1: Removed the dependency of the "DEBUG_EN" to enter the Debug mode. Updated Figure 973 to make PCI5_CH_CONFIG = "x" in the Normal state and PCI5_CH_CONFIG = "0" when moving from Config to Normal and also updated Table 1005 to reflect the changes. Change from MK Cut1: Removed the 3us constraint for switching among different modes(Normal-Disable-Config) Added Bullet-10 , in Table 1007 to show the reset value of GCR register dependency on the parameter PSI5_DIS_RST which is taken from the Device COnfiguration File. Updated the description of GLOBAL_DISABLE_REQ in Section 51.3.2.1, Global Control Register (PSI5_GCR) to show the reset value of GCR register dependency on the parameter PSI5_DIS_RST which is taken from the Device COnfiguration File Change from MK Cut1: Removed Debug_en bit from Section 51.3.2.2, PSI5 Channel Control Register (PSI5_PCCR) Updated the description of DSR_RDY, DBR_RDY and DPR_RDY in General Interrupt Status Register (PSI5_GISR). Updated the Bit 310 in Figure 989 , to "unwriteable" in test mode as this bit is the overwrite bit.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Peripheral Sensor Interface (PSI5) (continued) Updated the Bit 274 in Figure 990 , to “unwriteable” in test mode as this bit is the overwrite bit. Change from MK Cut1: Replaced the statement “As a corner caselater slot S1” in NOTES section after end of Table 1033 to “If S0 and S1 are...first slot(S0)” Added to the bit DRL description in Table 1034 , “..and optional messaging bits M0,M1” Added the statement “ As a special case if “S0” and “S1” are detected in different slots then NO T-bit error is set” in Section 51.3.2.29, Slot n End Boundary Register (PSI5_SnEBR) Change from MK Cut1: Replaced the statement As a special case if “S0” and “S1” are detected in different slots then NO T-bit error is set “ to “As a special case if “S0” and “S1” are detected in different slots then a T-bit error is set “. Change from MK Cut1: Added register bits and their descriptions , DATA_LENGTH[5:0] , CMD_TYPE[2:0] and SW_READY in Section 51.3.2.30, Data Output Block Configuration Register (PSI5_DOBCR) . Changed V13_EN bit name to DEFAULT_SYNC in Section 51.3.2.30, Data Output Block Configuration Register (PSI5_DOBCR) Updated the description of MDDIS_OFF in Section 51.3.2.31, Manchester Decoder Disable Offset Register (PSI5_MDDIS_OFF) Change from MK Cut1: Updated with the new jitter values , the NOTES section at the end of Table 1043 The whole description in Section 51.3.2.36, Data Preparation Register Low (PSI5_DPRL) updated to make it more clear , regarding the CRC dependence of the DPR[23:0] bits and the arrangement of data/address. Change from MK Cut1: Section 51.3.2.36, Data Preparation Register Low (PSI5_DPRL) updated to show the dependency of DPR[23:0] bits on CMD_TYPE[2:0] configuration. Updated NOTES section at end of Table 1044 Change from MK Cut1: Section 51.3.2.38, Data Buffer Register Low (PSI5_DBRL) updated to show the dependency of DBR[31:0] bits on CMD_TYPE[2:0] configuration. Updated NOTES section at end of Table 1046. The whole description of DBR[42:32] in Section 51.3.2.39, Data Buffer Register High (PSI5_DBRH) updated to make it more clear , w.r.t bit assignments . Change from MK Cut1: Section 51.3.2.39, Data Buffer Register High (PSI5_DBRH) updated to show the dependency of DBR[63:32] bits on CMD_TYPE[2:0] configuration. The DBR length increased to 64 bits. Updated NOTES section at end of Table 1047. The whole description of DSR[31:0] in Section 51.3.2.40, Data Shift Register Low (PSI5_DSRL) updated to make it more clear , w.r.t bit assignments . Change from MK Cut1: Section 51.3.2.40, Data Shift Register Low (PSI5_DSRL) updated to show the bit assignment of DSR[31:0] bits and it dependency on SW_READY bit . Updated NOTES section at end of Table 1048. The whole description of DSR[42:32] in Section 51.3.2.41, Data Shift Register High (PSI5_DSRH) updated to make it more clear , w.r.t bit assignments . Change from MK Cut1: Section 51.3.2.41, Data Shift Register High (PSI5_DSRH) updated to show the bit assignment of DSR[63:32] bits and it dependency on SW_READY bit . The DSR length increased to 64 bits. Updated NOTES section at end of Table 1049.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Peripheral Sensor Interface (PSI5) (continued) Added Bullet 2 to show the separate description of DPRL in Section 51.3.2.42, Device modes and Register bit accesses Section 51.3.2.42, Device modes and Register bit accesses also updated with other minor updates. Updated Figure 1020. Change from MK Cut1:Updated Section 51.4.3.4, Data transmission to include the complete support for standard length PSI5 frames and the non standard length frames.Updated Figure 1024 Change from MK Cut1:Updated Figure 1025 to include the enhanced ECU to Sensor communicatio support. Change from MK Cut1:Updates in Section 51.4.3.4.2, Data Preparation Register (64-bit register, DPRH + DPRL, DPRH is fixed at 0) and Section 51.4.3.4.3, Data Buffer Register (64-bit register) and Section 51.4.3.4.4, Data Shift Register (64-bit register). Updates in Section 51.4.3.4.5, Sync Pulse Trigger Generation and addition of "Programming and Control" bullet in the section. Updates in Section 51.4.3.4.5.1, Timing Pulses. Change from MK Cut1:Update in Figure 1026 to include the DEFAULT_SYNC bit and initialisation of all bits of DSR register to all "DEFAULT_SYNC" when DSR_RDY == 1. Change from MK Cut1:Update in Figure 1028 to show the control of GLOBAL_DISABLE_REQ bit also on _en(clock enables) of "rx" clock and "psi5" clock Change from MK Cut1:Update in 2nd bullet of "Jitter Related •" of Section 51.5.0.1, Latency and Jitter related requirements to show the jitter as "20ns". Change from MK Cut1:Updated Figure 1031 to show the updated jitters Added in the timestamp description in the data registers PSI5 Message Register High i (PSI5_PMRHi) {i: 0 to 31} etc. as "where "F" bit has been set ; the value of the Time Stamp appended to the message is "0" " Inserted Register sections in Section 51.4.6.1, Single DMA request for PSI5/diagnostic message buffers and Section 51.8.1, Internal SYNC Pulse generation and coordination across PSI5 channels Chapter SENT Receiver (SRX) No significant changes Chapter LINFlexD Added a note regarding baud rate under UART Mode features. Added a note in RDI field of LINSR register. Removed the Access and Reset value columns from Memory map table. Minor Editorial changes in the document Modified the LINS[3:0] fields of LINSR register from "w1c" to "read-only" Added table "Example of IBRR/FBRR and baud rate generated" under "Baud rate generation section" Updated note in "Baud rate generation section" Modified the description of DTU field from DISABLE timeout to RESET timeout in UARTCR register Removed line "PC[1:0] bit do not have any effect in parity" from 13-bit frames in "UART mode" section Modified "Example of IBRR/FBRR and baud rate generated" table

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Reset Generation Module (MC_RGM) Revised the definitions of the RGM_PRSTn registers. RGM_FBRE[BE_FCCU_HARD] reset value (was 0, is 1). RGM_FERD[D_JTAG_FUNC] bit type (was read/write, is read-only). Deleted the RGM_FEAR[AR_JTAG_FUNC] field (is reserved). Chapter Boot Assist Flash (BAF) Updated Table 10 Added soft DCF client, Secure watch dog, Callback and ADDR and DATAxx DCF client section Updated Figure 1 to incorporate DCF client Initialization. Chapter System Status and Configuration Module (SSCM) Added safety measures to Initialization and application information section Chapter Power Management Controller digital interface (PMC_dig) Significant updates to clarify design definition. Chapter Power Control Unit (MC_PCU) No significant changes Chapter Mode Entry Module (MC_ME) Register and field name changes: • PIT_RTI_0 to PIT_RTC_0 • DMA_CH_MUX0 to DMAMUX_0 • CRC_A0 to CRC_0 • ADCSD_A2 to ADCSD_4 • ADCSD_A1 to ADCSD_2 • ADCSD_A0 to ADCSD_0 • CAN_RAM_CTRL0 to CAN_RAM_CTRL • LINFlexD_A1 to LINFlexD_1 • LINFlexD_A0 to LINFlexD_0 • DSPI_A2 to DSPI_4 • DSPI_A1 to DSPI_1 • DSPI_A0 to DSPI_0 • IIC_A0 to IIC_0 • SENT_A0 to SENT_0 • FlexRay_A0 to FLEXRAY • PSI5_A0 to PSI5_0 • ADCSARB to ADCSAR_b • ADCSAR_A0 to ADCSAR_0 • CRC_B0 to CRC_1 • ADCSD_B2 to ADCSD_5 • ADCSD_B1 to ADCSD_3 • ADCSD_B0 to ADCSD_1 • LINFlexD_B0 to LINFlexD_2 • DSPI_B0 to DSPI_5 • DSPI_B1 to DSPI_3 • DSPI_B2 to DSPI_2 • SENT_B0 to SENT_1 • PSI5_B0 to PSI5_1

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Mode Entry Module (MC_ME) (continued) • ADCSAR_B4 to ADCSAR_6 • ADCSAR_B1 to ADCSAR_3 • ADCSAR_B0 to ADCSAR_2 In the "MC_ME register description" table, added entries for the following registers: • ME_PCTL31 • ME_PCTL85 In the "MC_ME register description" table, added requirement tags for the following registers: • ME_PCTL68 • ME_PCTL69 • ME_PCTL70 • ME_PCTL72 In the "MC_ME register description" table, deleted the following registers: ME_PCTL90 In the "MC_ME register description" table, added entries for the following registers: • ME_PCTL93 • ME_PCTL123 • ME_PCTL128 • ME_PCTL213 • ME_PCTL248 • ME_PCTL254 In the "MC_ME register description" table, deleted the following registers: • ME_PCTL126 • ME_PCTL164 • ME_PCTL219 • ME_PCTL250 • ME_PCTL251 Added the I_ICONF_CC field to the ME_IS register. Added the M_ICONF_CC field to the ME_IM register. Revised the definition of the ME_DMTS register. Added the S_PIT_RTC_1 and S_SIPI_0 fields to the ME_PS0 register. Added the S_LINFlexD_14 field to the ME_PS2 register. Revised the definition of the ME_PS3 register. Added the S_GTMINT field to the ME_PS4 register. Deleted the S_DMA_CH_MUX1 field from the ME_PS5 register. Revised the definition of the ME_PS6 register. Revised the definition of the ME_PS7 register. In the "MC_ME memory map" table, added fields to the ME_CS register (to align with the subsequent definition). Revised the definitions of the ME_CCTL registers. In the "MC_ME memory map" table, added entries for ME_CADDR1...4. Added the RMC field to the ME_CADDR registers. In the ME_RESET_MC and ME_DRUN_MC sections, added a note about chip configuration.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Mode Entry Module (MC_ME) (continued) Revised the field descriptions of the ME_CS register. Split the ME_CCTLn section into several sections, each describing one register. Split the ME_CADDRn section into several sections, each describing one register. In the “DRUN mode” section, added “The clock source configuration except system clock source 0 can be chosen by preloaded information in flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.” Revised the “SAFE mode” section. In the “Target mode request” section, deleted “including DRUN, RUN0...3, and TEST”. In the “MC_ME resource control overview” table, revised the entries for IRC and MVREG. In the “Peripheral clocks disable” section, added “ and the system clock frequency ramp-down of the System clock switching step”. Renamed “Processor and system memory clock disable” to “Processor clocks disable” and replaced the section's content. Deleted the “Additional Core Clocks Disable” section. Added the “System clock disable” section. Deleted the “Processor and Memory Clock Enable” section. Added the “System and processor clocks enable” section. Deleted the “Additional Core Clocks Enable” section. In the “System clock switching” section, deleted “if the value of the PWRLVL field of the ME_<target mode>_MC register is greater than that of the one of the ME_<current mode>_MC register,”. In the “Clock sources switch-off” section, added “For the clock sources related to the system clock”. In the “Flash memory switch-off” section, changed “This step is executed only when the Processor and System Memory Clock Disable process has completed” to “This step is executed only when the processor clocks disable and system clock disable processes have completed.” In the “Current mode update” section, changed “(halt/stop)” to “(stop)”. Replaced the Processor low-power mode exit” section. Chapter e200z4 Core Debug Support Split into two chapters—one for the e200z420n3 and one for the e200z425n3 Chapter Debug and Calibration Interface (DCI) Section 82.1.3, Operating modes updated for the usage of JTAGM_MCR[jtagm_JCOMP] when DCI is in DigRF mode Section 82.1.3.2, VSS detection updated to clarify the usage of LVDS fault condition not affecting operations of Nexus blocks. Corrected rev history Tagged Section 82.1.3.4, Nexus reset control with ADD23.138 (BLG_DCI_0046) Corrected the reference JTAGM_MCR[JCOMP] to JTAGM_MCR[jtagm_JCOMP] as used in JTAGM Block Guide Added tags to Synchronous restart control section Added bullet with tag (BLG_DCI_0047) to Break control section Added title to Enable Escape Mode field of DCI_CR

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter JTAG Controller (JTAGC) No significant changes
		Chapter IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG) No significant changes
		Chapter JTAG Data Communication (JDC) Updated Table 1310 to change JIN_WRITE opcode from 0x00011 to 0x01110.
		Chapter Sequence Processing Unit (SPU) No significant changes
		Chapter JTAG Master (JTAGM) Changed reset value of MCR from 0x0300_0002 to 0x0000_0002 Added a note in JTAM_MCR register Added section "JTAGM JCOMP usage in DigRF" Added table "Chapter type/source information" at the beginning of the BG
		Chapter DigRF - High Speed Debug No significant changes
		Chapter Development Tool Semaphore (DTS) No significant changes
		Chapter Nexus Aurora Router (NAR) Added NAR message formats table
		Chapter e200z4 Nexus 3 Module Split into two chapters—one for the e200z420n3 and one for the e200z425n3
		Chapter GTM Development Interface (GTMDI) Added section 1.5.9 Fetch trace Changed all references to NPC by NAR. Added DPLL and ARU Data Trace description in Features and Data Trace specific sections. Added Notes about timestamp meaning (no meaning) in Error Messages. Changed HWO bit description in Table 31 Fixed DPLL Data trace message Added Timestamp field description for Nexus messages Fixed conditional text Added note about GTM IP reset related to Fetch End message. Added a note after Figure 34 to explain fields are relative to MCS address map, not CPU map. Added a note after Figure 39 to explain fields are relative to MCS address map, not CPU map. Changed SRC value in Table 50 to 0x7 which is the usual SRC value for several devices (this is a variable that can be changed in the future). Added a footnote for Table 50 to explain FSCNT field. Added info about FSCNT in section 1.5.9.2/92. Added a note in section 1.5.9.2/92 about the MCS multi-thread behavior.

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Nexus Crossbar Multi-Master Client (NXMC) No significant changes Chapter Cyclic Redundancy Check Unit (CRC) Modified CRC unit memory map and registers. Chapter Memory Error Management Unit (MEMU) No significant changes Chapter Indirect Memory Access (IMA) All new content Chapter Fault Collection and Control Unit (FCCU) Removed Dual rail protocol (slow switching modes) diagram Removed Dual rail protocol (fast switching modes) diagram Removed time switching protocol diagram Removed Bi-stable protocol diagram Modified description of FCCU_SET_CLEAR field of FCCU_CFG register Changed all occurrences of FCCU_CFS0_3 to FCCU_CF_S0_3 Changed all occurrences of FCCU_NCFS0_3 to FCCU_NCF_S0_3 Changed all occurrences of FCCU_NCFE0_3 to FCCU_NCF_E0_3 for defect 8605 - Register naming leads to conflicts Replaced all occurrences of FCCU_NAFS with N2AF_STATUS Replaced all occurrences of FCCU_NFFS to N2FF_STATUS Replaced all occurrences of FCCU_NAFS with N2AF_STATUS Replaced all occurrences of FCCU_AFFS with A2FF_STATUS Replaced all occurrences of FCCU_FAFS to F2A_STATUS In FCCU AF Freeze Status Register (A2FF_STATUS) replaced SRC field with AF_SRC In FCCU NF Freeze Status Register (N2FF_STATUS) replaced SRC field with NF_SRC In FCCU FA Freeze Status Register (F2A_STATUS) diagram changed FAFS field from 9 bits wide to 8 bits wide Added a note in FCCU CF Fake Register (FCCU_CFF) description Added a note in FCCU NCF Fake Register (FCCU_NCFF) description In FCCU Transient Register (FCCU_TRANS_LOCK) diagram modified TRANSKEY field from 7 bits to 8 bits In FCCU Transient Register (FCCU_PERMNT_LOCK) diagram modified PERMNTKEY field from 7 to 8 bits Added a table "FCCU error pin behavior" under section "EOUT interface" Modified 12th bullet point in "Introduction" section Modified "EOUT interface" section Modified the description of FCCU_SET_CLEAR field of FCCU_CFG register Changed reset values of all the bits from "0" to "1" of FCCU_EOUT_SIG_EN0..3 register Added a note to the description of EOUTENx field of FCCU_EOUT_SIG_EN0..3 register Updated FCCU Output Supervision Unit (FOSU) section

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Self-Test Control Unit (STCU2) No significant changes Chapter Register Protection (REG_PROT) No significant changes Chapter ASC Modeling Unit No significant changes Chapter Wakeup Unit (WKPU) No significant changes Chapter GTM103 Integration (GTMINT) Module No significant changes Chapter Emulation Device Introduction No significant changes Chapter Buddy Device Crossbar and RAM No significant changes Chapter Buddy Device-System Integration Unit Lite (BD_SIUL2) No significant changes Chapter Buddy Device-Debug and Calibration Interface (DCI) Section 82.1.3, Operating modes updated for the usage of JTAGM_MCR[jtagm_JCOMP] when DCI is in DigRF mode Section 82.1.3.2, VSS detection updated to clarify the usage of LVDS fault condition not affecting operations of Nexus blocks. Corrected rev history Tagged Section 82.1.3.4, Nexus reset control with ADD23.138 (BLG_DCI_0046) Corrected the reference JTAGM_MCR[JCOMP] to JTAGM_MCR[jtagm_JCOMP] as used in JTAGM Block Guide Added tags to Synchronous restart control section Added bullet with tag (BLG_DCI_0047) to Break control section Added title to Enable Escape Mode field of DCI_CR Chapter Buddy Device-JTAG Controller (JTAGC) No significant changes Chapter Buddy Device - JTAG Master (JTAGM) Editorial updates throughout the document Removed "Access" and "Reset Value" columns from the Memory Map table. Added "FSL_Specific" and "ST_Specific" tags in Conditional text table. Added description of "non_cust" tag. Changed reset value of MCR from 0x0300_0002 to 0x0000_0002 Added a note in JTAM_MCR register Added section "JTAGM JCOMP usage in DigRF" Added table "Chapter type/source information" at the beginning of the BG

Table 2152. Document revision history (continued)

Date	Revision	Changes
01-Mar-2012	3 (continued)	Chapter Buddy Device-Nexus Aurora Router (NAR) Added NAR message formats table Chapter Buddy Device-Nexus Aurora Link (NAL) Added offsets to NAL memory map table Changed reset value of GSR bit 19 to one Bits 27–23 of TCR must be written with zeros. Chapter Buddy Device-Nexus Aurora PHY (NAP) No significant changes Chapter Buddy Device-Nexus Read Write Access (RWA) No significant changes Chapter Buddy Device - DigRF - High Speed Debug No significant changes
06-Jun-2013	4	Throughout – “MCAN” changed to “M_CAN” – “TTCAN” changed to “M_TTCAN” – DigRF replaced by LFAST (LVDS Fast Asynchronous Serial Transmission) – The combination of SIPI running on top of the LFAST interface is now referred to as Zipwire® – Removed ASC Modeling Unit (AMU) content Chapter Preface Following note added: “Any register bit described as “Reserved” should not be written. Reserved bits are not intended for customer use.” Section 1.1, Acronyms and abbreviations: – DTS definition changed to “Development Trigger Semaphore” (was “Development Tool Semaphore”) – FCCU definition changed to “Fault Collection and Control Unit” (was “Fault Control and Collection Unit”) Chapter Introduction Statement added: “For this reason, safety measures have been added which are further described in Chapter 6, Functional Safety, as well as the SPC57EM80xx Safety Manual”. Correction: I/O Processor (IOP) does not have internal data cache. Changes in features list: – Clarification of on-chip SRAM: “Up to 450 KB of total on-chip SRAM (128 KB general purpose SRAM, 192 KB data RAM, 48 KB instruction RAM, 35 KB GTM RAM, 16 KB overlay RAM, and 32 KB peripheral RAM)” – Item describing Boot Assist Flash support of factory programming via serial bootloader updated to include UART/LIN. – Correction: The SPC57EM80xx GTM103 has 34 KB of RAM (was 35 KB) – Specification change: CPU cores are not dual-issue – Added detail to on-chip SRAM item. – Changed Flash fetch accelerator description to, “Two 64-bit AHB ports, 256-bit read data interface. Each AHB port contains 4-entry, 2-way set-associative mini-cache”. (was “4 × 256-bit”)

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Introduction (continued) Changes to Block diagram figure: – Changed figure 1 to include “Triple INTC” and added Standby Regulator. Changes to Periphery allocation figure: – Changed “Peripheral Cluster A” to “Peripheral Cluster B”. – Changed “Peripheral Cluster B” to “Peripheral Cluster A”. – PS15_S_0 added to Peripheral Cluster B – Removed AMU – Added 32 KB RAM to Peripheral Cluster A Chapter Embedded Memories Extraneous flash layout removed from Figure 6 (Flash memory segmentation and read-while-write partitioning) Section 3.2.2.1, Standby voltage regulator deleted Section 3.2.2, Standby SRAM Regulator – First paragraph: changed “Standby SRAM functions....” to “The 64 KB standby SRAM on the MCU functions...” – 2nd paragraph replaced: “Change 2nd paragraph to: “The 64 KB standby RAM array is optionally powered by the VDDSTBY pin on the MCU. When the MCU core supply (VDD_LV) is powered down, the standby RAM contents can be maintained by supplying the VDDSTBY pin. See Section 59.7, Standby RAM Regulator Interface Logic, for more information on the VDDSTBY pin and the standby SRAM regulator.” – 3rd paragraph: Deleted (was “The standby regulator includes a switch to switch between core voltage and standby voltage; and a brown out detection circuit to detect when the standby supply has dropped below the RAM retention voltage”) – Clarification: There is 64 KB of standby SRAM – Details regarding brownout detection moved to PMC Digital Interface chapter Following sentence added to Section 3.2.3, Overlay SRAM: “Code can be executed from internal overlay RAM and extended overlay RAM where available. Execution from overlay RAM remapped over Flash can be enabled or disabled by use of PFCRCR[IRMEM]. Code can be executed directly from Extended Overlay RAM (address range beginning 0x0C000000) on the ED device or internal Overlay RAM (address range beginning 0x0D000000) regardless of the PFCRCR[IRMEM] setting.” – Section 3.3.3, Selecting flash blocks and all subsections deleted. Please see Chapter 37: Flash Memory Programming and Configuration for related content.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Signal Description <i>Note: I/O signal muxing information now appears in a Microsoft Excel® workbook attached to the device reference manual.</i> Section 4.4.1, Pin/ball startup and reset states: Table 6 (Pin/ball startup and reset states)—pin state changes: – TDI pin <i>startup state</i> is Weak pullup (was High-impedance) – TDO pin <i>State during reset</i> is High-impedance (was Weak pullup) – TMS pin <i>startup state</i> is Weak pullup (was High-impedance) – ESR0 <i>State during reset</i> is Output, strong pulldown (previously stated as Output, open drain) – Footnote clarification: “Analog input without output function—weak pull-up/weak pull-down can be activated only after reset sequence completion” – Footnote added: “F0 is also referred to as ERROR0 and FI0.” – Footnote added: “The SPC57EM80xx MCU does not implement a system configuration pin—TESTMODE is the only pin that affects device configuration. It is latched on the PORST rising edge and internally pulled-down.” Section 4.4.2, Power supply and reference voltage pins/balls: – Pin A20 moved from VSS_LV to VSS_HV. There is no VSS_LV on either the 176- or 216-pin packages – <i>Changed</i> $V_{DD_HV_JTAG_OSC}$ to $V_{DD_HV_IO_JTAG}$ – $V_{DD_HV_PMC}$ split into 2 rows: $V_{DD_HV_PMC}$ pin 153 in the QFP, dash for the BGA; second row is $V_{DD_HV_PMC_BYP}$ with dash for the QFP, and balls A19/B18 for the BGA – A1 added to VSS_HV for BGA292 PD and ED Section 4.4.3, System pins/balls: – Clarification of TESTMODE pin definition: “TESTMODE pull-down is implemented to prevent the device from entering TESTMODE. It is recommended to connect the TESTMODE pin to $V_{SS_HV_IO}$ on the board. The value of the TESTMODE pin is latched at the negation of reset and has no affect afterward. The device will not exit reset with the TESTMODE pin asserted during power-up.” (added detail regarding when TESTMODE pin value is latched and that device will not exit reset when pin is asserted during power-up) – Clarification on TESTMODE pin. Pin has an internal pull-down implemented to prevent the device from entering TESTMODE. Section 4.4.4, LVDS pins/balls: Table 11 (LVDSF pin descriptions): – Renamed to Aurora pin descriptions – LPBK_N signals deleted – LPBK_P signal deleted

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Signal Description Table 9 (LVDS pin descriptions): – Table title changed to “LVDS pin descriptions” (was LVDSM pin descriptions) – Pin numbers and ball coordinates added. – Changed “Interprocessor Bus LFAST” to “LFAST” – Title changed to “LVDS pin descriptions” (was LVDSM pin descriptions) – Pin numbers and ball coordinates added. – Functional block “Debug LFAST” changed to “High-Speed Debug (HSD) / LFAST” – Differential input pair added to DSPI 2: SIN_P (PH[7]) and SIN_N (PH[8]) – “_N” and “_P” suffixed deleted from DSPI signal names SCK, SIN, and SOUT – Correction: DEBUG_RXN is on port pin PA[5] (was PA[9]) – Correction: DEBUG_RXN is on pin 114 in the 176/216 package and L19 in the 292 package (was 113/H16) – Correction: DEBUG_RXP is on port pin PA[9] (was PA[7]) – Correction: DEBUG_RXP is on pin 113 in the 176/216 package and H16 in the 292 package (was 114/L19) – DEBUG_TXN is the “Debug LFAST, LVDS Transmit Negative terminal” (was incorrectly described as “...Positive Terminal”. – DEBUG_TXP is the “Debug LFAST, LVDS Transmit Positive terminal” (was incorrectly described as “...Negative Terminal”. – Added: DSPI 5 SIN LVDS pair (pos. is PI[15], neg. is PI[14]) – SIPI / LFAST changed to Zipwire / LFAST Section 4.4.5, Generic pins/balls: – Removed I/O Signal Description table and replaced with reference text: See the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. Chapter Memory Map Introduction to memory map section expanded. Table 11 (Overview Memory Map): – 0x00A00000 to 0x00FFFFFF region description changed from “Small & Medium Flash Blocks...” to “Low and Mid Flash Blocks...” – Clarification: End address of UTEST NVM block changed to 0x00403FFF – Used Size for UTest NVM block is 16 KB – Clarification: BAF flash block added – Clarification: Reserved space added after BAF flash block – TTCAN changed to M_TTCAN – MCAN changed to M_CAN Table 12 (Peripheral (PBRIDGE_A, PBRIDGE_B) Memory Map): – AMU content deleted – SIPI module (SIPI_1) added (0xFFFD4000 to 0xFFFD7FFF) – PS15_S (PS15 UART Module) added (0xFBF74000 to 0xFBF77FFF) – Two columns added to indicate which PBRIDGE/slot each peripheral is mapped to

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Memory Map (continued) Table 13 (Flash memory and overlay RAM map): – Added “Address Space” column Table 15 (RAM memory map): – Correction: Standby RAM address range is 0x4000_0000 to 0x4000_FFFF. System RAM address range is 4001_0000 to 0x4001_FFFF. Table 16 (UTest flash memory map): – Reserved space at address 0x00400004 extended to 0x0040000B – Reserved spaces at addresses 0x004000C4 and 0x004000C8 combined – Area from 0x00400200 through 0x004002FF is now occupied by Lifecycle functions followed by reserved space (was part of DCF records). – Area from 0x00400300 through 0x00400307 is now the DCF Start Record – Area from 0x00400308 through 0x00400347 is now DCF HSM 'ROM' key Records – Area from 0x00400348 through 0x00400FFF contains remainder of DCF records – Area from 0x00404000 through 0x00407FFF is reserved for BAF code – Description of temperature sensor calibration constants updated Chapter Functional Safety Throughout: – PBRIDGE_0 is now PBRIDGE_A Section 6.4.1.1, End-to-End protection on data path: Removed the sentence “This is particularly true with peripherals connected to the I/O bridges.” - Updated last two sentences in paragraph tagged SCG17.001 in section “End-to-End protection on data path” to “This setup leaves the data path downstream of the ECC units (for example, starting at, and downstream of, the peripheral bridges) unprotected by ECC. To detect corruptions introduced on those unprotected data paths, the PBRIDGES and peripherals can be used redundantly (see Section 6.5.1.3, I/O peripherals), or other mechanisms may be used to validate peripheral operation.” Section 6.5.1.2.1, Clock: – Changed CMU_11 to CMU_1 as appears in Chapter 27, Clocking

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Functional Safety (continued) Section 6.5.1.4.1, Disabling of communication controllers: - Updated the second paragraph in section "Disabling of communication controllers" to read "When the FCCU intends to drive FI[0] to a fault condition, the SIUL2 disables the output buffers of all pins whose SIUL2_MSCRn[SMC]=0. This disables the possible transmission of erroneous messages until the FCCU intends to drive FI[0] to a non-fault condition. Once the communication controller transmission ports are disabled, they remain in the disabled state as long as the FCCU is driving a fault condition on FI[0]." - Clarification: "As long as the FCCU drives FI[0] to a fault state, the communication controller's Tx pins are disabled. As soon as FI[0] is driven to a non-fault state, the communication controller's Tx pins are automatically enabled." - Updated description of automatic disabling of output drivers Added NOTE: "Disabling of communication controllers occurs on the Tx pins of the communication controller (for example, CAN, FlexRay) of the MCU even while the communication controller continues operating. Failed transmissions, when detected by the communication controller, may lead to additional error events". Section 6.5.1.5.2, Online Logical BIST (LBIST) Added "Not in LBIST partitions" and "LBIST partitions" tables. Section 6.5.1.9, FCCU supervision (FOSU): - Clarification: If the FCCU does not react within a predefined interval as specified in the "FCCU configuration" section of the "Device configuration" chapter, on one of those outputs to the incoming failure indication, the FOSU assumes the FCCU has failed, and causes a <i>destructive</i> reset. ("destructive" added) Section 6.5.2, Dual-core lockstep: - Updated LSM description Section 6.4.4.1, ECC for storage: - Changed "ECC RAM implementations" table column name from "Halts on uncorrectable error" to "Reacts to uncorrectable error" - Renamed Core_0 (Main Safety) to Core_0(Safety Core) (Master and Checker cores) in "ECC RAM implementations" table - Renamed Core_0 - Checker to Core_1 in "ECC RAM implementations" table - Renamed Core_1 to Core_2 in "ECC RAM implementations" table Table 21 (FCCU failure inputs): Table headings changed: - Changed column heading 'FI[0] indication default at POR' to 'Source Enabled at POR' - Changed heading 'Source' to 'Failure'. Contents of column unchanged except where noted below. - Changed heading 'Signal(s)' to 'Failure description'. Detailed fault description replaces previous content. - New column: 'Reference Manual related information' contains links to relevant content of reference manual. FCCU channel 0 (TEMP_ERROR): - Changed 'Failure description' (formerly 'Signal(s)'): "Event indication from temperature sensor. The fault is cleared by clearing the status in MEMU_EPR_TD, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Temperature detector")

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Functional Safety (continued) Table 21 (FCCU failure inputs): FCCU channel 1 (LVD_ERROR): – Changed 'Failure description' (formerly 'Signal(s)'): "Voltage out of range indication from LVDs. The fault is cleared by clearing the status in MEMU_EPR_VDn, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Voltage out of range from LVDs.") FCCU channel 2 (HVD_ERROR): – Changed 'Failure description' (formerly 'Signal(s)'): "Voltage out of range indication from HVDs. The fault is cleared by clearing the status in MEMU_EPR_VDn, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Voltage out of range from HVDs") FCCU channel 3 (DPMC_DCF_SAFETY_ERR): – Changed 'Failure description' (formerly 'Signal(s)'): "Digital PMC initialization error during DCF data load. The status gets cleared if the fault is not persistent." (was "Digital PMC DCF Safety Error") FCCU channel 4 (MEMORY_DCF_SAFETY_ERR): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication of fault during timing/repair configuration initialization by SSCM of different memories. The status is cleared by clearing SSCM_STATUS[CER], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]. The status gets cleared if the fault is not persistent." (was "Memory Timing/Repair DCF Safety Error") FCCU channel 5 (SSCM_XFER_ERR): – Changed 'Failure description' (formerly 'Signal(s)'): "SSCM transfer error (during the SSCM and STCU DCF loading). The fault is cleared by clearing SSCM_STATUS[CER], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "SSCM transfer error (during the SSCM and STCU DCF loading)") FCCU channel 6 (STCU_UF): – Changed 'Failure description' (formerly 'Signal(s)'): "STCU2 unrecoverable fault indication during self-test. Fault is cleared by clearing STCU_ERR_STAT[UFSF] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "BIST result—wrong signature (STCU unrecoverable fault)") FCCU channel 7 (STCU_RF): – Changed 'Failure description' (formerly 'Signal(s)'): "STCU2 recoverable fault indication during self-test. Fault is cleared by clearing STCU_ERR_STAT[RFSF] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "BIST result - wrong signature (STCU recoverable fault)") FCCU channel 8 (STCU_LMBIST_USR_ERR): – Changed 'Failure description' (formerly 'Signal(s)'): "Activation of LBIST or MBIST control during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "STCU Fault in case LBIST or MBIST control signals go to wrong condition during user application.") Channel 9 (SPURIOUS_DEBUG_SSCM_ACTIVATION): – New Failure (formerly 'Source') name is SPURIOUS_DEBUG_SSCM_ACTIVATION) (was "JTAG_NPC_MON")

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Functional Safety (continued) Table 21 (FCCU failure inputs): FCCU channel 9 (SPURIOUS_DEBUG_SSCM_ACTIVATION): Changed 'Failure description' (formerly 'Signal(s)'): "JTAG, NPC or Debug functionality out of reset or SSCM activation during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "JTAG or NPC not in reset or activations of dangerous debug functionality") FCCU channel 10 (RCCU_0): – Changed 'Failure description' (formerly 'Signal(s)'): "Safety cores out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "rcc_out") FCCU channel 11 (RCCU_1): – Changed 'Failure description' (formerly 'Signal(s)'): "Safety cores out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "rcc_out") FCCU channel 12 (SWT2_ResetReq): – Changed 'Failure description' (formerly 'Signal(s)'): "Reset request from Software Watchdog of Peripheral Core_2 domain" (was "Software watchdog timer2 Reset Request") FCCU channel 13 (SWT1_ResetReq): – Changed 'Failure description' (formerly 'Signal(s)'): "Reset request from Software Watchdog of Peripheral Core_1 domain" (was "Software watchdog timer1 Reset Request") – Changed 'Default reaction...': "No reaction" (was "Long Reset") – Changed 'Source enabled at POR': "No" (was "Yes") FCCU channel 14 (SWT0_InterrupReq): – Changed 'Failure description' (formerly 'Signal(s)'): "First timeout interrupt request from Software Watchdog of Safety Core (Main Core_0)" (was "LBIST + SW test by not triggering SWT during startup") FCCU channel 15 (SWT0_ResetReq): – Changed 'Failure description' (formerly 'Signal(s)'): "Second timeout (reset) request from Software Watchdog of Safety Core (Main Core_0)" (was "Software watchdog timer of Safety Core, second timeout (reset)") FCCU channel 16 (MEMU_RAM_CE): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication by MEMU of occurrence of correctable errors in system RAMs. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "MEMU RAM correctable error") FCCU channel 17 (MEMU_RAM_UCE): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication by MEMU of occurrence of uncorrectable errors in system RAMs. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "MEMU RAM uncorrectable error") FCCU channel 18 (MEMU_RAM_BOV): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication of overflow of system RAM ECC buffer in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "MEMU RAM buffer overflow")

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Functional Safety (continued) Table 21 (FCCU failure inputs): FCCU channel 19 (MEMU_PER_CE): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication by MEMU of the occurrence of correctable errors in peripheral RAMs. The fault is cleared by the clearing MEMU_ERR_FLAG[PR_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]" (was "MEMU peripheral correctable error") FCCU channel 20 (MEMU_PER_UCE): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication by MEMU of the occurrence of uncorrectable errors in peripheral RAMs. The fault is cleared by the clearing MEMU_ERR_FLAG[PR_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]" (was "MEMU peripheral uncorrectable error") FCCU channel 21 (MEMU_PER_BOV): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication of overflow of system RAM ECC buffer in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[PR_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]" (was "MEMU peripheral buffer overflow") FCCU channel 22 (MEMU_FLS_CE): – Changed 'Failure description': "Indication by MEMU of occurrence of correctable errors in flash memory. The fault is cleared by the clearing MEMU_ERR_FLAG[F_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "MEMU flash memory correctable error") FCCU channel 23 (MEMU_FLS_UCE): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication by MEMU of occurrence of uncorrectable errors in flash memory. The fault is cleared by the clearing MEMU_ERR_FLAG[F_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "MEMU flash memory uncorrectable error") FCCU channel 24 (MEMU_FLS_BOV): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication of overflow of flash memory ECC buffer in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[F_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "MEMU Flash buffer overflow") FCCU Channel 25 (IMA_SOC): – Changed 'Failure' (formerly 'Source'): "IMA_SOC" was "IMA_R") – Changed Failure description' (formerly 'Signal(s)'): "MCU IMA active" (was "IMA read access") FCCU Channel 26 (IMA_HSM): – Changed 'Failure' (formerly 'Source'): IMA_HSM (was "IMA_W") – Changed Failure description' (formerly 'Signal(s)'): "HSM IMA active" (was "IMA write access") FCCU channel 27 (SARB_WD): – Changed 'Failure description' (formerly 'Signal(s)'): "Reference-SAR ADC watchdog is triggered" (was "Reference-SAR watchdog") FCCU channel 28 (SARn_WDG): – Changed 'Failure description' (formerly 'Signal(s)'): "OR'd outputs of watchdogs, SAR_ADC[0:7]" (was "SAR[0:7] watchdog (OR'd signals)")

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Functional Safety (continued) Table 21 (FCCU failure inputs): FCCU channel 29 (FM_PLL_0): – Changed 'Failure description' (formerly 'Signal(s)'): "PLL0 Loss of lock (interrupt). Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "PLL0 Loss of lock (interrupt)") FCCU channel 30 (FM_PLL_1): – Changed 'Failure description' (formerly 'Signal(s)'): "PLL1 Loss of lock (interrupt). Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "PLL1 Loss of lock (interrupt)") FCCU channel 31 (CMU_0_OSC): – Changed 'Failure description' (formerly 'Signal(s)'): "Loss of XOSC clock. The fault is cleared by clearing CMU_ISR[OLRI] of CMU_0, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Loss of XOSC clock") FCCU channel 32 (CMU_0_PLL): Changed 'Failure description' (formerly 'Signal(s)'): "SYSCLK frequency out of range. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_0, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]" (was "SYSCLK frequency out of range") FCCU channel 33 (CMU_comp_subsys): – Changed 'Failure description' (formerly 'Signal(s)'): "Exception from monitoring internal clocks of comp_subsys. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_n, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]" (was "Monitoring internal clocks of comp_subsys") FCCU channel 34 (CMU_other): – Changed 'Failure description' (formerly 'Signal(s)'): "Exception from monitoring internal clocks of other subsystems. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_n, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]" (was "Monitoring internal clocks of other subsystems") FCCU channel 35 (PERIPH_XBAR Transaction Monitor): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication of a hardware fault resulting in corrupted transaction through the peripheral XBAR. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Peripheral XBAR Transaction monitor mismatch") FCCU channel 36 (EDC_ECC_FLASH): – Changed 'Failure description' (formerly Signal(s)): "Indication of a hardware fault in the flash memory resulting in corrupted ECC detection/correction. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "EDC after ECC for flash memory Array") FCCU channel 37 (EDC_ECC_FLASH_C): – Changed 'Failure description' (formerly Signal(s)): "Indication of hardware fault in the flash memory controller resulting in corrupted ECC detection / correction. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "EDC after ECC for flash memory Controller")

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Functional Safety (continued) Table 21 (FCCU failure inputs): FCCU channel 38 (ENC_ERR_FLASH): – Changed 'Failure description' (formerly Signal(s)): "Indication of hardware fault resulting in corrupted flash memory access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Flash Encoding Error") FCCU channel 39 (ADDR_FDBK_ERR_FLASH_C): – Changed 'Failure description' (formerly Signal(s)): "Indication of hardware fault in the flash memory controller resulting in corrupted flash memory access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "PFlashC Address Feedback Error") FCCU channel 40 (COMP_XBAR_Transaction_Monitor): – Changed 'Failure description' (formerly Signal(s)): "Indication of hardware fault resulting in corrupted transaction through the computational XBAR. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Computational XBAR Transaction monitor mismatch") FCCU channel 41 (ON_CAL_RAM_ERR): – Changed 'Failure description' (formerly Signal(s)): "Indication of hardware fault in flash memory controller resulting in corrupted access to on-chip overlay RAM. Status should be cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "On-chip Calibration RAM Feedback Error") FCCU channel 42 (SAFE_CAL_ERR): – Changed 'Failure description' (formerly Signal(s)): "Indication of hardware fault in the flash memory controller resulting in corrupted overlay remap evaluation. Status should be cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Safe Calibration Error") FCCU channel 43 (ADDR_FDBK_ERR_PRAM + RAM_LWB_ERR): – Changed 'Failure description' (formerly Signal(s)): "Indication of hardware fault in system RAM controller resulting in corrupted RAM access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "System RAM Address Feedback + RAM Late-Write Buffer mismatch") FCCU channel 44 (MPU_ERROR): – New FCCU channel 45 (EDC_ERR_PRAM): – Changed 'Failure description' (formerly Signal(s)): "Indication of hardware fault in the system RAM controller resulting in corrupted ECC detection/correction. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "EDC after ECC for RAM (read-modify-write ECC)") FCCU channel 46 (DFT1): – Changed 'Failure description' (formerly 'Signal(s)'): "Test circuitry Group 1 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Test circuitry Group 1 activation") FCCU channel 47 (DFT2): – Changed 'Failure description' (formerly 'Signal(s)'): "Test circuitry Group 2 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Test circuitry Group 2 activation")

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Functional Safety (continued) Table 21 (FCCU failure inputs): FCCU channel 48 (DFT3): – Changed 'Failure description' (formerly 'Signal(s)'): "Test circuitry Group 3 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]. (was "Test circuitry Group 3 activation") FCCU channel 49 (DFT4): – Changed 'Failure description' (formerly 'Signal(s)'): "Test circuitry Group 4 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Test circuitry Group 4 activation") FCCU channel 50 (Safety Core Machine Check Exception): – Changed 'Failure description' (formerly 'Signal(s)'): "Safety Core Machine Check Exception" (was "Safety Core Exception") – Added note that begins "See 'Exceptions and Conditions' table in the <i>e200z4 Core Reference Manual</i> for..." FCCU channel 51 (Lockstep mode): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication of disabled Checker Core or RCCUs. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Indication of disablement of Checker Core or RCCUs") FCCU channel 52 (Safe_Mode): – Changed 'Failure description' (formerly 'Signal(s)'): "Indication of safe mode request assertion from MC_RGM. The event is also captured in MC_ME and available as status for crosscheck. Status should be cleared first in the MC_RGM_FES, then followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Safe Mode Entry Indication") FCCU channel 53 (Compensation Disable): – Changed 'Failure description' (formerly 'Signal(s)'): "Pad Compensation Disabled Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Pad Compensation Disabled") FCCU channel 54 (EIN_ERR): – Changed 'Failure description' (formerly 'Signal(s)'): "Error input pin (FI[0], external of the MCU)The fault is cleared by clearing the fault driving FI[0] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]." (was "Error input pin (from the external world)") – Added table footnote: "EIN_ERR can be used only when bistable protocol is selected for external error indication." FCCU channel 55 (SWT3_ResetReq): – Changed 'Failure description' (formerly 'Signal(s)'): "Reset request from Security Watchdog" (was "Software watchdog timer3 Reset Request") Section 6.4.5, Register classifications: – Content removed because it is still under development. Will be added in a future revision.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Device Configuration Section 7.2, Core modules – Details about locations of core modules added to introduction Section 7.3.7.3, Interrupt sources – IRQ 671 (SPU_INTS[INT]) deleted – IRQ 226 PIT1 Channel 0 PIT_1_TFLG0[TIF] changed to PIT0 Channel 0 PIT_0_TFLG0[TIF] – IRQ 227 PIT1 Channel 1 PIT_1_TFLG1[TIF] changed to PIT0 Channel 1 PIT_0_TFLG1[TIF] – IRQ 228 PIT1 Channel 2 PIT_1_TFLG2[TIF] changed to PIT0 Channel 2 PIT_0_TFLG2[TIF] – IRQ 229 PIT1 Channel 3 PIT_1_TFLG3[TIF] changed to PIT0 Channel 3 PIT_0_TFLG3[TIF] – IRQ 230 PIT1 Channel 4 PIT_1_TFLG4[TIF] changed to PIT0 Channel 4 PIT_0_TFLG4[TIF] – IRQ 231 PIT1 Channel 5 PIT_1_TFLG5[TIF] changed to PIT0 Channel 5 PIT_0_TFLG5[TIF] – IRQ 232 PIT1 Channel 6 PIT_1_TFLG6[TIF] changed to PIT0 Channel 6 PIT_0_TFLG6[TIF] – IRQ 233 PIT1 Channel 7 PIT_1_TFLG7[TIF] changed to PIT0 Channel 7 PIT_0_TFLG7[TIF] – IRQ 239 PIT1 RTI Channel PIT_1_RTI_TFLG[TIF] changed to PIT0 RTI Channel PIT_0_RTI_TFLG[TIF] – IRQ 240 PIT0 Channel 0 PIT_0_TFLG0[TIF] changed to PIT1 Channel 0 PIT_1_TFLG0[TIF] – IRQ 241 PIT0 Channel 1 PIT_0_TFLG1[TIF] changed to PIT1 Channel 1 PIT_1_TFLG1[TIF] – Deleted IRQ 481 Loss-of-clock interrupt request for PLL0 PLL0SR[LOCF] – Deleted IRQ 483 Loss-of-clock interrupt request for PLL1 PLL1SR[LOCF] – Interrupt sources 494 and 495 added – Interrupt source 831 (GTM error) added – Interrupt sources 946–961 (PSI5-S error) added Updated source descriptions and source names for IRQs 484–485: – IRQ 484—was ipi_int_ext_pd1_done / ipi_int_ext_pd1_done; is PLL0 External Power Down Cycle Complete indication interrupt / PLL0SR[EXTPDF] – IRQ 485—was ipi_int_ext_pd2_done / ipi_int_ext_pd2_done; is PLL1 External Power Down Cycle Complete indication interrupt / PLL1SR[EXTPDF] Deleted IRQ 486 (ipi_int_pll1_clk_sw) Deleted IRQ 487 (ipi_int_pll2_clk_sw) Section 7.3.9, DMACHMUX configuration, Table 43 (DMAMUX peripheral DMA request to input source mapping): – Changed DMA_CH_MUX_3 source 46 from “DSPI_1_TX” to “DSPI_1_RX” and source 47 from “DSPI_1_RX” to “DSPI_1_TX” – Added source: DMA_CH_MUX_2: 49 (PSI5-S) – Added source: DMA_CH_MUX_3: 50 (PSI5-S)

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Device Configuration (continued) Section 7.4, Clocking – Added cross-reference to Clock Monitor Unit (CMU) chapter Section 7.5.1.1, Calibration remapping support – New section Section 7.5.1.2, Wait state settings for overlay flash latency matching – New section added Section 7.6.1, Temperature sensor configuration – “temperature sensor bandgap voltage” changed to “ADC bandgap voltage” Section 7.7.2, Software Watchdog Timer (SWT) configuration, Table 55 (Software Watchdog Timer (SWT) instances): – Replaced description of SWT3. – Added “Internal 16 MHz RC oscillator clock that is used for timer operation” to features list Section 7.7.2.3, Default configuration: – Added brief explanation of how Security Watchdog is enabled and cross-reference to BAF chapter. – added details on SWT default configurations per SWT_CR register reset values. Section 7.8.2, FEC interface selection, Table 62 (FEC interface selection): – Revised to reflect the correct method for selecting interfaces using RCR[MII_MODE] and SIUL_SCR0[FEC_MODE]. – Footnote added: MII supported only in BGA packages Section 7.8.5.4, High Speed and MicroSecond Channel (MSC) Support: – There are four package input pins to be muxed onto any of the 32 DSPI parallel inputs (was two input pins) – Listed pins that can be inserted into the MSC datastream: PC[2], PD[8], PF[3], and PJ[4]. – Added bit 28 “XSPI to Figure 25 (DSPI Module Configuration Register (DSPIx_MCR))). – Note added: DOZE mode is not implemented Section 7.8.6.2, Stopwatch timer input: – New section Section 7.9.1.2, Device specific features, Section Table 73., SSCM_MEMCONFIG Reset Values: – Footnotes added to emphasize that values will change with each revision of the device. – Added Table 74 (SSCM startup information addresses) and introductory sentence, “Table 74 shows the address locations that the SSCM uses to search for startup information.” Section 7.10.2.5, MEMU registers, Table 83 (Corresponding error sources for FLASH OFLW register): – Revised the address offsets to align with the main MEMU content. – Added the PERIPH_RAM_OFLW0 register. – Changed FLASH_OFLW to FLASH_OFLW0 for consistency in naming with the other OFLW registers.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Device Configuration (continued) Section 7.10.3.1, SRAM Arrays accessible by processor cores: – Specified processors that can access SRAM arrays – Cross-reference to IMA chapter added Section 7.10.3.1, SRAM Arrays accessible by processor cores, Table 86 (SRAM arrays accessible via IMA): – GTM FIFO ECC bits changed to 35–29 (was 36–30) – GTM MCS RAM0 and MCS RAM1 ECC bits changed to 38–32 (was 38–33) – GTM DPLL RAM1a and RAM1b ECC bits changed to 30–24 (was 31–25) Section 7.10.4.2, FOSU parameters: – Replaced 'safe clock' with 'IRCOSC' Section 7.10.5, Register protection (REG_PROT) configuration: – Added Section 7.10.5.14, JTAG Master (JTAGM) protected registers Table 89 (Reference links to related information) – Deleted Power Control Unit (MC_PCU) register(s) from Table 89 (Reference links to related information). Section 7.10.5.1, SIUL2 protected registers: – SIUL2_IFER0, SIUL2_IFMC0-SIULIFMC31, and SIUL2_IFCPR registers deleted – MSCR512-543, MSCR576-587, MSCR608-612, MSCR614-615, MSCR619, MSCR624-628, MSCR630-631, MSCR638-645, MSCR656-719, MSCR752-754, MSCR758-760 registers added – MCSR1023 register deleted Section 7.10.5.2, Core Clock Monitor Unit (CMU_CORE) protected registers: – CMU section split into two sections—one for the core CMU and another for the I/O processor CMU Section 7.10.5.4, PMC Digital Interface (PMC_DIG) protected registers: – Registers listed individually instead of summarized, e.g., REE_VD3 and REE_VD4 instead of REE_VDn. – FEE_VD3, FEE_VD4, FEE_VD7, FEE_VD9, FEE_VD10, FEE_VD12, FEE_VD13, FEE_VD14, FEE_VD15 registers added – ADC_CH register added – CTL_TDn register renamed to CTL_TD – VD_UTST register added – Deleted NT_TDn registers Section 7.10.5.5, PLL Digital Interface (PLL_DIG) protected registers: – PLL0_CR and PLL0_DIVR registers added – PLL2CR, PLL2DV, PLL2FM, and PLL2FD registers deleted – PLL1DV register renamed to PLL1_DIVR – PLL1CR register renamed to PLL1_CR Section 7.10.5.10, Clock Generation Module (MC_CGM) protected registers: – Clarification: Byte 3 of CGM_AC[0..15]_SC registers are protected Section 7.10.5.11, Reset Generation Module (MC_RGM) protected registers: – RGM_STDBY register deleted Section 7.10.5.12, System Status and Control Module (SSCM) protected registers: – Error_Config register renamed to ERROR – Protected size of ERROR changed to 16 bits (was 8)

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Device Configuration (continued) Section 7.11.1.2, PASS Module DCF Records (includes life cycle state and censorship): – Figure 28 (PASS_LOCK0_PGn Register) and related tables: footnote added detailing where to find more information. – Added ATSL bit to Figure 28 (PASS_LOCK0_PGn Register). Chapter Reset and Boot Section 8.4.3.7, Locate HSM Boot Header (state 44): – Correction: List of locations checked by SSCM for a valid boot header changed to 0x60_C000, 0x61_0000, and 0x62_0000 (was 0x60_0000, 0x60_4000, and 0x61_4000). Section 8.3.3, PHASE0 Phase: analog supply initial configuration: Correction: Flash memory is not initialized during PHASE0 and the flash memory module does not have a self test capability. Chapter Device Configuration Format (DCF) Records Table 112 (DCF client list): – Added STCU_MB_CTRL registers 70 thru 77 – Changed the Tamper Detect Diary Base Address reset value for the DCF client to “0xFFFFFFFF”. – Changed the Tamper Region Override for DCF client special strategy to “Once+Write ‘1’ only”. – Changed TDR0_LOCKx for DCF client special strategy to “Write Once+Write ‘1’ only”. – Changed Miscellaneous - UTEST Miscellaneous, Reset value of DCF client to “0x00000941” – Corrected Reserved space ranges for TDM DCF clients – Corrected CS bits for PMCDIG REE bits client – Corrected Reserved space ranges for TDM DCF clients Table 113 (UTEST Miscellaneous DCF Client field descriptions): – Changed bit 19 and bit 20 to “RESERVED”. Chapter Power Management Table 120 (Functional terminal state during power-up and reset): – Changed TMS POWERUP pad state from “high impedance” to “weak pull-up” – Changed TDI POWERUP pad state from “high impedance” to “weak pull-up” – Changed TDO RESET pad state from to “weak pull-up” to “high impedance” Chapter Security Editorial updates Chapter Calibration and Debug Section 12.3.2, Debug LFAST: – Added sentence “In addition, the LFAST switch allows direct access to read and write memory.”

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Core Complex Overview Section 13.2, Features: – Correction: e200z425n3 core has 64 KB DMEM (was 32 KB) Section 13.3.8, Local memory features: – Correction: e200z425n3 core has 64 KB DMEM (was 32 KB) Section 13.3.7, Cache features: – Clarification: DBL1CCSR0 register provides hardware debug cache invalidate support for data cache. – CDANTL and CDADATA registers provide software and hardware debug access to cache tag, status, and data storage. Chapter Core e200z420n3 description Editorial changes Chapter Core e200z425n3 description Editorial changes Chapter System Integration Unit Lite2 (SIUL2) Register name correction: Changed occurrences of "Multiplexed Single Configuration Register" to "Multiplexed Signal Configuration Register". Register mnemonic (MSCR) is unchanged. Table 225 (SIUL2_MIDR2 field description): – Updates made to FLASH_SIZE_1 memory sizes in description column. Table 235 (SIUL2_MSCR0 - SIUL2_MSCR511 field description): – Updated "OERC" field description column. – INV bit (16), added note: "Use of the INV bit is not supported when the ODC is configured for open-drain or open-source modes, and the output data will not be as expected for an inverted open-drain or open-source connection." – Added note: Not all pads have all bits i.e., only some pads have hysteresis, see the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. – Changed cross reference from Chapter 4, I/O Signal Description table, to the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. – Added note to OERC field description: OERC[2] programming is required for forward compatibility. – Updated WPDE bit description: Weak Pullup Enable—Used only when the associated destination is a chip pin. Enables the associated pin's weak pullup resistor. It is OK for both WPDE and WPUE to be enabled, if they are, the pad is neither a weak pullup and a weak pulldown. – Updated WPUE bit description: Weak Pulldown Enable—Used only when the associated destination is a chip pin. Enables the associated pin's weak pulldown resistor. It is OK for both WPDE and WPUE to be enabled, if they are, the pad is neither a weak pullup and a weak pulldown.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter System Integration Unit Lite2 (SIUL2) (continued) Section 14.3.2.16, Module-port MSCR assignments and SSS values – Removed reference text and Table 271. Peripheral and input multiplexing assignments. Replaced with: For each internal module port that is or can be configured as an input and has more than one possible source. See the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the Input Multiplexing Table tab. The Input Multiplexing Table shows the assigned MSCR, and the SSS values and their corresponding source signals. Section 16.2.2.17, SIUL2 GPIO Pad Data Output Registers (SIUL2_GPDO0_3–SIUL2_GPDO508_511): – Removed: See the Chapter 4, Signal Description for mapping of ports to GPDI registers. Replaced with: For mapping of ports to GPDO registers, see the SPC57EM80xx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. Table 16.3.4.1 (External interrupt initialization) – Added “(For DMA, its self clearing)” to following item: ““Write to EIFn bits in DISR0 to as desired to clear any flags Chapter Crossbar Switch (XBAR) Editorial changes Chapter Crossbar Integrity Checker (XBIC) Section 18.1, Overview: Detail added regarding new functionality: – Forward signals (master to slave) crossbar transfer integrity is checked by verifying the attribute information using an 8-bit Error Detection Code (EDC). – During the data phase of the system bus pipelined protocol, the XBIC compares the three signals of hready, hresp0, hresp2 from entering the crossbar on the slave side to leaving the crossbar on the master side. Data phase is often more than one clock cycle due to wait cycles. All three signals are compared clock by clock during the whole data cycle Section 18.2, Features: – New feature: Verification of feedback information for each data phase during crossbar transfer. – New feature: Programmable integrity check enable on a per-master-port basis Section 18.5.2.1, XBIC Module Control Register (XBIC_MCR): – The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis. (previously stated, “...to be enable on a per-slave-port basis.” – Field description changed: Slave Port Enable (SEn) field description changed to “Slave Port Enable for EDC error detect” – Fields added: Master Port Enable for slave driven signal safety check (ME0–ME7)

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Crossbar Integrity Checker (XBIC) (continued) Section 18.5.2.3, XBIC Error Status Register (XBIC_ESR): – Functional description changed: The Error Status Register (XBIC_ESR) contains two types of error information about the last transfer. If an attribute integrity check error was detected, the slave port, the master port and the syndrome are captured in the SLV, MST and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are captured in the DPSE0-7 and DPME0-7 fields. – Fields added: Data phase slave port error (DPSE0-DPSE7) – Fields added: Data phase master port error (DPME0-DPME7) – Minor editorial updates Section 18.6, Functional description: Detail added: – The above for XBIC to catch forwarding (master to slave) signal errors during AHB address phase. XBIC also detect backward signal (slave to master) errors during AHB data phase. – For the three slave driven signals (hready, hresp0 and hresp2), XBIC perform the comparison between the input of the slave port and output of the master port during the AHB data phase. When mis-match happening, the error will be captured. The current slave port will be registered in XBIC_ESR bit field DPSE0-7, and the current master port will be registered in XBIC_ESR bit field DSME0-7. Chapter Crossbar Locking Control (CLC) Editorial changes Chapter Peripheral Bridge (PBRIDGE) Editorial changes Chapter System Memory Protection Unit (SMPU) Clarified the maximum allowed size of regions. Clarified that access control permissions are defined per region descriptor, not globally. Section 21.4.2.7, Region Descriptor n, Word 2 Format 0 (SMPUX_RGD_WORD2_FMT0) – Clarification: “RGD_WORD2_FMT0 defines the access control rights of the memory region” to “...of the memory region <i>on a per master basis</i>” Chapter Intelligent AHB Gasket (IAHBG) Added new sections on 64->32 and 32->64 interfaces Chapter Semaphores2 (SEMA42) Section 23.2, Memory map/register definition: – Corrected the representation of RSTGT (is one register at offset 0x0040). – Added the following to the RSTGSM field of the SEMA4_RSTGT register. “The “10” state persists for only one clock cycle. Software will never be able to observe this state.” – Modified Semaphores (Secure) Reset Gate (SEMA4_RSTGT) note 2 to state that write sequence errors will cause reset sequence to abort.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Interrupt Controller (INTC) Section 24.4.1, Software vector mode: - Deleted “Furthermore, the interrupt vector to the processor is driven as all zeroes. The interrupt acknowledge signal from the associated processor is ignored.”. Section 24.4.2, Hardware vector mode: - Deleted the paragraph beginning with “The processor negates the interrupt request...”. Section 24.5, Memory map and register definition: - Deleted “IPS” from the note. - Added new programmable register to the memory map, Master Protection Register (INTC_MPROT). This register is used to protect cores from writing to other cores INTC registers. Adding this register impacts other programmable registers in the chapter which now rely on INTC_MPROT. Section 24.5.2.7, INTC Priority Select Registers (INTC_PSR0 – INTC_PSR1023): - Removed the note, “The PRC_SELn or PRIn field of an INTC_PSRn must not be modified while its corresponding peripheral or software-settable interrupt request is asserted.” and the note about area optimization. Section 24.5.2.7, INTC Priority Select Registers (INTC_PSR0 – INTC_PSR1023): - Revised Table 279 (INTC memory map) and to represent the fact that each PSR is 16 bits wide, containing a PRC_SEL and PRI field. In Section 24.6.1.3, Unique vector for each interrupt request source, rewrote the last sentence to include the correct starting peripheral interrupt request number. Chapter Enhanced Direct Memory Access (eDMA) Editorial changes Chapter DMACHMUX No changes Chapter Clocking Throughout: - Removed all references to CGL (Clock Gating Logic). It is an internal module not visible to the customer Section 27.3, System clock frequency limitations: - Removed NOTE, “HSM_CLK and PBRIDGEEx_CLK are not required to be at the same frequency, but the COMP_CLK, CHKR_CLK, FXBAR_CLK, and IOP_CLK must always be at the same frequency.” Section Figure 263., Device Ethernet clocking: - Corrected diagram to show the MSCR in the SIUL module, and not the MC_CGM.A - Added the sentence, “When a 40MHz oscillator is used as the input clock to the FlexRay controller, both edges of the clock are used by the module to achieve the required 80 MHz frequency.” - Corrected diagram to show the MSCR in the SIUL module, and not the MC_CGM.A

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Clocking (continued) Section 27.6.5, FlexRay clocking: – Added the sentence, “When a 40MHz oscillator is used as the input clock to the FlexRay controller, both edges of the clock are used by the module to achieve the required 80 MHz frequency.” – Added sentence, “Selection of PLL or external oscillator as the FlexRay clock is configured in the MC_CGM (see Figure 350 (MC_CGM auxiliary clock 2 generation overview) in Chapter 30: Clock Generation Module (MC_CGM))” Figure 258 (Clock generation (part 1 of 2)): – Added footnote “The clock trees COMP_CLK and CHKR_CLK are isolated after System Clock Divider x. This isolation is maintained in the physical design of the clock trees on the device.” to diagram and enclosed effected area referencing footnote – Removed DSPI_6 – Added footnote “The clock trees COMP_CLK and CHKR_CLK are isolated after System Clock Divider x. This isolation is maintained in the physical design of the clock trees on the device.” to diagram and enclosed effected area referencing footnote – Added Figure Footnote “DSPI_CLK1 and DSPI_CLK0 are the protocol clocks as described in the “Deserial Serial Peripheral Interface (DSPI)” chapter.” – Moved PSI5 clock tree details to separate diagram. Details have received significant updates. Figure 259 (Clock generation (part 2 of 2)): – New diagram contains details of PSI5 clock tree Table 340 (MC_CGM relationship to clocks): – Removed DSPI_6 from 'MC_CGM relationship to clocks', CGM_AC0_DC3 row, Output clock column. Section Figure 262., Device LFAST clocking: – Correction: TCK/DRCLK maps to PA[6] (was PA[5]) Section 27.3, System clock frequency limitations: – Removed NOTE, “HSM_CLK and PBRIDGEEx_CLK are not required to be at the same frequency, but the COMP_CLK, CHKR_CLK, FXBAR_CLK, and IOP_CLK must always be at the same frequency.” – Clock dividers added Section 27.7.1.2, CMU registers and field availability: – Removed CMU_ISR[FLCI] from Table 350 (CMU registers availability) 'CMU registers availability'. Figure 261 (Clock distribution): – PSI5-S added Figure 268 (Loss of clock logic): – Interrupt connection from CMU_0 to INTC deleted

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Clocking (continued) Table 340 (MC_CGM relationship to clocks) New clock dividers added: – CGM_AC5_CDC10 – CGM_AC5_CDC11 – CGM_AC5_CDC12 – CGM_AC5_CDC13 – CGM_AC5_CDC20 – CGM_AC5_CDC21 – CGM_AC5_CDC22 – CGM_AC5_CDC23 Table 342 (Maximum system level clock frequencies) New clocks and clock dividers added: – CDC10/PSI5_S_DDSR0_CLK – CDC11/PSI5_S_DDSR1_CLK – CDC12/PSI5_S_DDSR2_CLK – CDC13/PSI5_S_DDSR3_CLK – CDC2/PSI5_S_WD_CLK – CDC20/PSI5_S_WD_CLK0 – CDC21/PSI5_S_WD_CLK1 – CDC22/PSI5_S_WD_CLK2 – CDC23/PSI5_S_WD_CLK3 Section 27.6.2, PSI5 clock dividers: Frequencies for new clocks added: – F_{PSI5_S_TS_CLK} – F_{PSI5_S_DDSR0_CLK} – F_{PSI5_S_DDSR1_CLK} – F_{PSI5_S_DDSR2_CLK} – F_{PSI5_S_DDSR3_CLK} – F_{PSI5_S_WD0_CLK} – F_{PSI5_S_WD1_CLK} – F_{PSI5_S_WD2_CLK} – F_{PSI5_S_WD3_CLK}

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Dual PLL Digital Interface (PLLDIG) Section 28.6.2, Clock configuration: – Updated NOTE that starts “The PLL must be powered down and powered up in the MC_ME...” to remove the unclear text. – Updated equations Equation 4, Equation 5, Equation 8 and Equation 10 to show particular relationship of each equation contents to PLL0, PLL1 or both. – Added equations Equation 6, Equation 7 and Equation 12. – Equation 5: Added which shows relationship between PLL1 input (fpll1_ref) and PLL1 output (fpll1_phi). – Equation 5: Updated by adding divide by 2. – Equation 7: Updated due to the update to Equation 5. – Equation 8: Added which shows relationship between PLL1 input (fpll1_VCO) and PLL1 output (fpll1_phi). – Equation 10: Updated Eqn 8 by changing PLL0DV[RFDPHI] to PLL0DV[RFDPHI] – Equation 8: Removed 'x 2' from each equation. Section 28.4, Modes of operation: – Added note “Mode changes are controlled by configuring the MC_ME Mode Configuration registers (MC_ME_<mode>_MC).” – Added the bullet “Bypass mode with reference, and PLL1PLL0either PLL0PLL1 or both PLLs enabled” as one of the modes of operation. Section 28.5.2, Register descriptions: – Removed the dimensioned registers PLLnCR, PLLnSR and PLLnDV, and replaced them with the non-dimensioned versions, PLL0CR, PLL1CR, PLL0SR, PLL1SR, PLL0DV and PLL1DV. Chapter Clock Monitor Unit (CMU) Editorial updates Chapter Clock Generation Module (MC_CGM) Table 430 (MC_CGM CGM_PCS_DIVCn[RATE] values): – Multiplied the values in the “d = CGM_PCS_DIVCn[RATE]” column by 1000. – Deleted entries for $a_{max} = 0.25$ and higher. Chapter OSC Digital Interface (XOSC) Section 31.3.1.1, XOSC control register (XOSC_CTL): – Updated reset value for M_OSC to ‘–’ and added note referring customers to the “Clocking” chapter. Chapter RCOSC Digital Interface Section 32.1, Functional description: – “During power on reset sequence, native trimming bits are initialized from the TEST flash memory” is updated to read “During power on reset sequence, native trimming bits, fine trimming bits, and TSENS code bits are initialized from the TEST flash memory”. Section 32.2.1.2, Native trimming register (NT): – Bits 2-7 are updated from “Reserved” to “FT Code”. – Bits 10:15 are updated from “Reserved” to “TX Code”.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter RAM Controller (PRAM) Throughout: minor editorial updates. Table 441 (RAM controller memory map) - Table footnote updated to indicate that accessing PRCR1 register outside of supervisor mode results in a transfer error. Section 33.3.1.2, Optional read wait-state: "...access will take 2T to complete..." changed to "...access will take 2 clock cycles (2T) to complete..." "3T" changed to "3 clock cycles (3T)" Section 33.3.1.2, Optional read wait-state: - Following note added: "The number of cycles taken for a RAM access can vary ± 1 clock cycle depending on the RAM speed relative to the PRAM controller clock frequency. If system RAM is running at the same frequency as the PRAM controller a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to PRAM controller clock frequency." Section 33.2.1, Platform RAM Configuration Register 1 (PRAMC_PRCR1): Section Table 442., Platform RAM Configuration Register 1 description in - Following footnote added: "The number of cycles taken for a RAM access can vary ± 1 clock cycle depending on the RAM speed relative to the PRAM controller clock frequency. If system RAM is running at the same frequency as the PRAM controller a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to PRAM controller clock frequency." Section 33.2.1, Platform RAM Configuration Register 1 (PRAMC_PRCR1): - Field deleted: Port p1 read buffer disable (P1_RB_DIS) - Field deleted: Port p0 read buffer disable (P0_RB_DIS) - Field added: Port p0 read burst optimization disable (P0_BO_DIS) - Field added: Port p1 read burst optimization disable (P1_BO_DIS) - Field deleted: Read One Wait State (RD1WS) - Field added: Flow through disabled (FT_DIS) - Note added: "This register is accessible only in supervisor mode. Accesses in user mode returns a transfer error." Section 33.3.1.2, Optional read wait-state: - First paragraph revised subsequent to deleting PRCR1[RD1WS] field and adding PRCR1[FT_DIS] field. Changed, "The controller can be optionally programmed to insert one additional wait-state in the AHB response on reads. Setting the PRCR1[RD1WS] field inserts additional one wait state in the AHB dataphase on read requests. The state of PRCR1[RD1WS] field has no effect on write." to "The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCR1[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCR1[FT_DIS] field has no effect on writes."

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Flash Memory Controller Throughout: – Editorial changes – Formerly each AHB port contained a 4-way set associative mini-cache. The mini-cache has been changed to 2-way. Section 34.1, Introduction: – Added item to list of functions: “It acts as the interface between the ASC Modeling Unit (AMU) and the c55fm flash memory array.” Section 34.1, Introduction: – Detail added to feature item. The two AHB ports allow simultaneous access to dedicated prefetch mini-cache per slave port. Section 34.4.1, Overview: – Added detail to statement that attempted updates to the programming model while the flash memory controller is in the midst of an operation will result in non-deterministic behavior: “Software must be architected to avoid this scenario. – Added: “There is no idle indicator for the flash controller. The recommended flow for multi-core devices is to start only one core and execute initialization code to completion before starting the remaining cores. If the user needs to reconfigure the flash, code execution must be temporarily moved to system RAM.” Section 34.4.1.1, Platform Flash Configuration Register 1 (PFCR1): Changes to PFCR1[P0_BFEN] function: – Deleted: “This bit is cleared by hardware reset.” – Added: “This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset.” Section 34.4.1.2, Platform Flash Configuration Register 2 (PFCR2): Changes to PFCR2[P1_BFEN] function: – Added: “This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset.” Section 34.5.6, Flash error response operation: – “or upon detection of an ECC event” deleted from following sentence: “The flash array may signal an error response to terminate a requested access due to improper sequencing during program/erase operations, improper sequencing during array integrity testing or upon detection of an ECC event.” Section 34.5.9, Access pipelining: – Following note added: “Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.”

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Flash Memory Controller (continued) Section 34.5.14, PFlash calibration remap support: – Added: “Timing for calibration accesses to overlay RAM is only guaranteed if RWSC is set sufficiently high to cover the intrinsic burst access incurred when fetching a full page of calibration data” – Deleted: Note that when a reset event asserts while an access to the on-chip overlay RAM is requested, the flash controller aborts the array access to ensure that RAM contents are not corrupted.” – Deleted: “Since the calibration access is defined to have the same access time as a normal flash array access, remapped references to the system RAM are treated as the highest priority request by the platform’s RAM controller.” – “for a flash access as well as the required access time” deleted from statement: “Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency.” – Deleted: “The translation function is performed during the first AHB data phase cycle.” Section 34.5.16.2.2, e2eECC and Calibration Accesses: – Clarification and corrections to example of where contents of CRD contents are redundantly defined. Chapter Embedded Flash Memory (c55fmc) Table 456 (Block Memory Map): – Correction: Register LOCK2 has R/W access (was previously listed as read-only) – Correction: Register LOCK3 has R/W access (was previously listed as read-only) – Correction: Register SEL2 has R/W access (was previously listed as read-only) – Correction: Register SEL3 has R/W access (was previously listed as read-only) Section 35.2.1.1, UTest NVM block: – Test mode disable seal content moved to new section Section 35.2.1.2, Test Mode Disable Seal – New. Content was previously included in Section 35.2.1.1, UTest NVM block. – Test mode disable value is 0x2D3C_4B5A (was incorrectly stated as 0x5A4B_3C2D). – Clarification: “Once sealed, the Utest block becomes OTP (Erase Locked, and Over Program Protection enabled). The Utest Block also is not accessible in Test Mode” – Clarification: “Utest operations Margin Read and Array Integrity are also protected by preventing MISR updates on blocks selected for protection, although reads will still be done and Single Bit Corrections and Double Bit Detections will be logged.” – Added: “In order to let two opportunities to select blocks for the Test Mode Disable, two regions are available, Test Mode Disable Block Select - 0 and Test Mode Disable Block Select - 1.”

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Embedded Flash Memory (c55fmc) (continued) Section 35.2.2, Register Summary, Section Table 459., Register summary: – Register added: TMD Section 35.2.3, Register descriptions: – Detail added to field descriptions in SELn registers. – New fields (CPR, CPA, CPE) added to UT0 register – Clarification: UT0[AIE] remains at 1 when the Array Integrity Check, or Margin Read are suspended or are in a breakpoint, and cannot be cleared. – Clarification: "Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way." Chapter Decorated Storage Memory Controller (DSMC) Minor editorial changes Chapter Configuring Flash Memory Write and Read Protection Completely rewritten Chapter Analog-to-Digital Converters (ADC) Configuration Section 38.2.2.7.4, Analog input pin switch to VSS_HV_ADV: – Figure 451 (Analog input AN35 — discharge switch to AVSS): pull up/down pad cell removed – "The ground switch for AN7/AN35 is controlled in input multiplexing MSCR registers in the SIUL2." changed to "...MSCR registers (MSCR[WPDE]) in the". Figure 411 (ADC subsystem): – Added lines between SARADC instances and GTM. Section 38.2.1.2, Sigma-Delta ADC analog input multiplexing: – SARADC_B analog inputs are switchable by the SIUL2_MSCRn[APC] bit. – SDADC analog inputs have no SIUL2_MSCRn[APC] bit control, but are directly connected to the pad. Table 507 (SAR ADC test channel register descriptions (SARB only)): – Changed SARADC_B input channel 108 to "Trimmed PMC bandgap reference" – Changed VAGND1 to VSS_HV_ADV (ground supply for ADC) – Changed Vssm to VSS_HV_ADR_S – Changed Vddm to (VDD_HV_ADR_S - VSS_HV_ADR_S) Chapter Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface Section 39.7.12, Offset calibration support: – Step ⁽²⁾ ("Configure the mux selection ANCHSEL field of CSR to '100' or '101' as required") expanded with two sub-steps. – Editorial updates

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface <i>Section 40.5.1.40, External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–3):</i> – The register name and offset address above the figure has been corrected. Changed from “TCCAPR0–7: 0x050–0x05CC” to “ECMICR0–3: 0x5C0–0x05CC”. <i>Figure 464 (SARADC block diagram):</i> – Footnote added: “Test channels are mapped to internal analog channels via the SARADCB_TCCAPRx registers” – Added pointers to functional equivalent of TCCAPR7 register fields ICSEL_TCH0, ICSEL_TCH1, ICSEL_TCH2, and ICSEL_TCH3 <i>Section 40.5.1.1, Main Configuration Register (MCR):</i> – Changed reset value of WTRGOUT bit <i>Section 40.4.5, Test channel connection with internal analog channel:</i> – Updated the TCCAPRX register description for BigEndian format Editorial changes Chapter Temperature Sensor Editorial changes Chapter Timers Overview Added footnote to GTM103 Wrapper Module Configuration in “Timers” table. Chapter System Timer Module (STM) No changes Chapter Software Watchdog Timer (SWT) Editorial changes Chapter Periodic Interrupt Timer (PIT) Editorial changes Chapter CAN Subsystem – Added a note in the M_TTCAN BTP register – Added a note under M_CAN Dual clock sources – Added a note under M_TTCAN Dual clock sources – Minor editorial changes – Added Example code in the M_CAN Software initialization – Added a new feature as a note below M_CAN features section – Added two new notes below M_CAN and M_TTCAN Dual clock sources respectively. – In the M_CAN CCCR register, corrected the access of TEST bit as RW. – In the M_CAN TOCV register, corrected Bits 16:31 name to TOC. Corrected bit field description. – In the M_CAN IR register, corrected HPM bit map description. – In the M_CAN IR register, corrected RF1L bit map description.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter CAN Subsystem (continued) – In the M_CAN RXF0C register, corrected bit access to RW. – In the M_CAN section, added a new section "Filtering for Debug Messages". – In the M_TTCAN RWD register, corrected Bits24:31 name to WDC. – Updated M_CAN Block diagram – Updated Rx handler description – M_CAN Registers Description: added FBTP, NDAT1,NDAT2, RXBC registers, updated TEST, CCCR, BTP, PSR,IR,IE,ILS, RxF1S – Updated Message RAM section – Update Standard and Extended Message ID Filter Element bit field descriptions – Functional Description: Added a new section "CAN FD Operation", updated Acceptance filtering features, added a new section "Filter for specific IDs", added a new section "Dedicated Rx Buffers", added a new section "Debug on CAN Support" – Updated M_CAN message memory size requirement table – Updated M_TTCAN feature list, Rx handler description, Message RAM Configuration – Updated M_TTCAN Rx Buffer and FIFO Element, Tx Event FIFO Element, – Added M_TTCAN CAN FD Operation. – Updated M_TTCAN Timeout counter, Rx Handling, Acceptance filtering – Added M_TTCAN Dedicated Rx Buffers section – Corrected M_CAN TOCV register reset. – Converted M_CAN and M_TTCAN PSR[LEC] bit map to binary – Corrected M_TTCAN BTP registers field description table – Added M_TTCAN_RXF1S[DMS] bit – Added M_TTCAN_IR[DRX], M_TTCAN_IE[DRXE], M_TTCAN_ILS[DRXL] – Added M_TTCAN_RXF1S[DMS] – Corrected M_CAN_IR[DRX], M_CAN_IE[DRXE], M_CAN_ILS[DRXL] bit access as "w1c" – Updated M_CAN[CCCR] register – Added few cross-references in the M_CAN and M_TTCAN registers. – Added M_TTCA_TTMLM[CSS], M_TTCA_TTOCN[TTIE] – M_TTCAN Extended message ID filter element section : Replaced EIDFC with XIDFC – M_TTCAN: Renamed Rx FIFO element section with Rx Buffer and FIFO element section – Updated M_CAN FD Operation – Updated M_TTCAN FD Operation – Updated "Hardware control for M_TTCAN cycles management" figure – Replaced MCAN with M_CAN and TTCAN/TT_CAN with M_TTCAN throughout the document. – Section 46.3.5.2.11, Timeout Counter Value Register (TOCV): Corrected the register access from "R/W" to "W1C". – Section 46.3.5.2.25, Rx FIFO 0 Configuration Register (RXF0C): Corrected the FOS bit field length from [10:15] to [9:15] – Section 46.3.5.2.26, Rx FIFO 0 Status Register (RXF0S): Added new bits FOPI[10:15]

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter CAN Subsystem (continued) – Section 46.3.5.2.30, Rx FIFO 1 Status Register (RXF1S): Added new bits F1PI[10:15] – Section 46.3.5.2.32, Tx Buffer Configuration Register (TXBC): Corrected the register access from “R” to “RP”. – Section 46.3.5.2.33, Tx FIFO/Queue Status Register (TXFQS): Added new bits TFGI[19:23] – Section 46.3.5.2.42, Tx Event FIFO Status Register (TXEFS): Added new bits EFGI[19:23] Chapter Serial Interprocessor Interface (SIPI) Section 47.2, Overview: – Following added to end of section: “The IPS clock and HCLK(AHB interface clock) for SIPI are of the same frequency and synchronous.” – Updated Section 47.14.1.8, SIPI Module Configuration Register (SIPIMCR) – Added new bit “CHNSB” in Section 47.14.1.8, SIPI Module Configuration Register (SIPIMCR) – Replaced table “SIPI header channel number coding” with Section Table 759., SIPI Header Channel Number Coding. Section Figure 689., SIPI module: – “Channel 2 is given the highest priority for transmission, and is the only channel that is capable of data streaming” is updated to read “Channel 0 is given the highest priority for transmission, and channel 2 is the only channel that is capable of data streaming”. Section 47.14.1.1, Channel Control Register (CCRn): – “channel 2 – channel 0 – channel 1 – channel 3” is updated to read “channel 0 – channel 1 – channel 2 – channel 3”. Section 47.14.1.1, Channel Control Register (CCRn): – The “note” text is updated to read “ This register is only writable in Initialization mode (MCR[INIT] = 1), except for command type bits ie. TC, WL, ST, IDT, RRT and WRT. Also, the command bit types can be written only when corresponding Channel busy bit, CB, is not asserted so SW has to poll the respective channel busy bits before trying to change the command type. At a time only one channel can transmit so only one channel can remain 'busy' and maximum time needed to transmit a full command is 9 clock cycles. Once a channel is not busy then its Command Type can be changed. Also, to send a command, we need to write to the corresponding CARn register(address register)”. Section 47.14.1.1, Channel Control Register (CCRn): – The description of bit “CHEN” read “If all the channel enables are enabled simultaneously, channel 2 will be given the highest priority for transmission. For channels 0, 1 and 3, the lowest channel number is given highest priority.” Here Channel 2 is now Channel 0 and references to Channels 0, 1 and 3 has been removed.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Deserial Serial Peripheral Interface (DSPI) – Modified Async Clocking Scheme and CMDTCF Feature Descriptions. – Updated descriptions of following bit fields - MCR[MDIS], CTAR[FMSZ], SR[CMDTCF], SR[TFFF], PUSH[PE_MASC], PUSH[PP_MCSC], DSICR0[FMSZ5], DSICR0[TSBC], DSICR1[TRGPRD], DSICR1[DPCSx]. – Updated register description for DSPI_PUSH_Slave, DSPI_RXFRx – Changed CTAR_SLAVE[FMSZ6] to CTAR_SLAVE[FMSZ[5]]. – Updated description within DSI Transfer Initiation Control, DSI Serialization, Modified Transfer Format, Interleaved TSB (ITSB) Mode. – Updated mention of system clock to protocol clock at all places appropriate. Chapter DigRF - Interprocessor Communications Table 848 "LCR field description": – Added text "The internal loopback is not intended for board level functionality, so this feature should not be implemented." to the end of the field description of LCR[LVLPE]. Section 49.6.2.16, Unsolicited Tx Control Register (UNSTCR): – Updated field name for UNSTCR[25:31] to UNSHDR Section 49.7.6.2.2, Unsolicited frame receive flow: – Updated UNSRSR[USXDV] to UNSRSR[URXDV]. – Updated RISR[TXUNSF] to RISR[RXUNSF], bullet 3. Section 49.6.2.24, PLL and LVDS Status Register (PLLSR): – Table 857, 'PLLSR field descriptions'. Swapped fields LDSLPS and LRSPLS. Section 49.6.2.15, LVDS Control Register (LCR): – Updated reset value of the LVRXOP field to 100b. Reset value of LCR is now 0000_5024h. Chapter Fast Ethernet Controller (FEC) Section 50.4, Memory map and register definition: – Added "Do not modify the content of reserved fields." Re-architected the register information to present the MIB block counter registers as "real" registers (each register is presented in the overall memory map and has a separate section with the typical register figure and field description). In Section 50.4, Memory map and register definition: – Deleted "You control the FEC by writing into control registers located in each block." – Added "All accesses to and from the registers must be via 32-bit accesses. There is no support for accesses other than 32-bit." In Section 50.4.21, Receive Descriptor Ring Start Register (ERDSR): – Added "You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions." – Changed "This register is not reset" to "This register is undefined at reset". – Changed all RMON and IEEE registers from read-write to read-only. Chapter FlexRay Communication Controller (FlexRay) Editorial changes

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Inter-Integrated Circuit (I2C) In Figure 1115 (I2C transaction protocol), changed “XX” to “Data”. In Figure 1121 (I2C high-level protocol steps), added “Acknowledgment from slave” after step 2. In Figure 1122 (Slave address transmission process), in step 4, changed “The master must generate a STOP condition or a repeated START condition” to “The master must initiate a data transfer followed by either a STOP condition or a repeated START condition”. In Section 52.6.5, DMA application information, changed “is configured in master mode” to “is configured in slave mode”. In Table 1108 (Chip modes supported by the I2C module), deleted the entry for DOZE mode (not supported on this chip). In Section 52.4.6, I2C Bus Control Register (IBCR), deleted the IBDOZE field (that bit position is reserved). Section 52.2.4, Features: Added following footnote to features list: “Compliant with I2C 2.0 standard with the exception that HS (high speed) mode is not supported” Chapter Peripheral Sensor Interface (PSI5) Changed the Jitter values to “end of last CRC bit of message” in Section 53.5.1, Latency and Jitter related requirements. Corrected the position of the last stuff bit (in CRC field) of “long” and “xlong” frames in Table 1188 (Standard Frame Formats Supported). Updated Table 1131 (PSI5 port list) to change the pad names to SDINn_PSI5_m and SDOUn_PSI5_m. Replaced “ipp_sdin” to “sdin” and “ipp_sdout” to “sdout” throughout the doc. – Updated Table 1172 (Start Bit Detection) and Table 1173 (Bit Extraction). – Updated Section 53.4.2.3.4.10, Message Too Early (in first slot and overlaps Manchester Decoder disable) and Section 53.4.2.3.4.11, Message Too Late (overlaps next message) to replace 32 samples of idle bit duration by 42 samples. Section 53.3.2.1, Global Control Register (PSI5_GCR): – Changed “Device Configuration File” to “Device Configuration Chapter” in two places. Table 1132 (PSI5 memory map): – Changed “Device Configuration File” to “Device Configuration Chapter” in note 4 at end of table. Removed the statement “DND == 1, if the FIRST_TRANSITION_DETECT occurs and $43 \leq \text{psi5_clk32x_idle_detect_cnt} \leq 45$ and invalid SECOND_TRANSITION_DETECT” from Table 1172 (Start Bit Detection) Updated 1ms in 5th bullet of Figure 1195 (Sync Pulse Generation Parameters and Jitters) to 1us. Table 1129 (Normal operating states): Under “Cycle Time”, “Number to time slots per cycle” is modified to read “Number of time slots per cycle”. Chapter Peripheral Sensor Interface UART interface (PSI5-S) New chapter

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter SENT Receiver (SRX) In Section 55.3.2.15, DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3), some of the LSB bits in the Field Descriptions table were always visible; conditioned all to be LSB. In Section 55.3.2.20, Channel 'n' Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG), some of the LSB bits in the Field Descriptions table were always visible; conditioned all to be LSB. – In Section 55.3.2.5, Slow Serial Message Ready Status Register (SMSG_RDY), first paragraph, third sentence: changed “register will get asserted on Fast Message reception” to “register will get asserted on Slow Message reception.” – In Section 55.4.8, Time stamp logic, corrected cross-reference to STATUS register by changing its target to CTRL register. – In Section 55.3.2.4, Fast Message Ready Status Register (FMSG_RDY), first paragraph, third sentence: changed “register will get asserted on Slow Message reception” to “register will be asserted on Fast Message reception.” – In Section 55.3.2.5, Slow Serial Message Ready Status Register (SMSG_RDY), first paragraph, third sentence: changed “register will get asserted on Fast Message reception” to “register will be asserted on Slow Message reception.” Chapter LINFlexD Updated Section 56.2.2, UART mode features In Table 1268 (UARTCR field descriptions) updated description of CSP field, updated description of OSR field and changed field “RSE” to “ROSE” Added a bullet point in Section 56.2.1, LIN mode features Replaced “reset” with “set” in “slave mode” in Section 56.3.2.5.5, Overrun error Updated DRBNE bit from “read only” to “w1c” in Section 56.4.2.3, LIN Status Register (LINSR) Changed the reset value of SLEEP field from “0” to ‘1’ in Section 56.4.2.1, LIN Control Register 1 (LINCR1) Added field “AUTOSYNC_COMP” in Section 56.4.2.3, LIN Status Register (LINSR) Updated the description of DTU field in Section 56.4.2.5, UART Mode Control Register (UARTCR) Updated the description of LSIE field in Section 56.4.2.2, LIN Interrupt enable register (LINIER) Updated Figure 1317 (Master node — DMA Tx FSM (concept scheme)) Editorial updates Changed the description of signals “ipg_stop” and “ipg_stop_ack” in Table 1292 (Interface signals) Updated description of Section 56.4.2.24, UART Current Timeout Register (UARTCTO) Updated description of Section 56.4.2.23, UART Preset Timeout Register (UARTPTO) Updated description of WL0, WL1, WLS and DTU fields in Section 56.4.2.5, UART Mode Control Register (UARTCR) Updated description of Noise flag field in Section 56.4.2.6, UART Mode Status Register (UARTSR) Updated Figure 1291 (Block diagram) Updated Figure 1319 (Master node — DMA Rx FSM (concept scheme)) Updated Section 56.3.2.5.2, Bit error Updated bit description of NF field in Section 56.4.2.6, UART Mode Status Register (UARTSR)

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter LINFlexD (continued) Updated Section 56.3.2.2, Test mode Updated Section 56.2.1, LIN mode features Updated Section 56.4.2.23, UART Preset Timeout Register (UARTPTO) and Section 56.4.2.24, UART Current Timeout Register (UARTCTO) Added MIS field to Section 56.4.2.5, UART Mode Control Register (UARTCR) Section 56.2.2, UART mode features: – Features list item, “1/2 stop bits” changed to, “1/2/3 stop bits” Section 56.2.2, UART mode features: Features list item: “For bit rate $\leq$ 5 Mbit/s / Sixteen times oversampling / 3:1 majority voting” changed to, “For bit rate $\leq$ 12.5 Mbit/s...” Section 56.2.2, UART mode features: Features list item: “For 5 Mbit/s < bit rate $\leq$ 25 Mbit/s / Reduced over sampling programmable by the user / 1:1 voting” changed to “For 12.5 Mbit/s < bit rate $\leq$ 25 Mbit/s / Reduced over sampling programmable by the user / 3:1 majority voting for reduced over sampling of 8 / 1:1 voting for all reduced over sampling of 4, 5 and 6” Changes made throughout chapter for 25 Mbit/s support. Section 56.3.2.2, Test mode Clarification added to Loopback Mode: The TX pin can be disconnected from LINTX pin by SIUL2 setting Section 56.3.2.5.2, Bit error: Calculation of allowable transceiver delay expanded. Section 56.3.2.8, Baud rate generation: Sentence rewritten to note that $ipg_baud_clk = f_{CPU} = LIN_CLK$. Section 56.4.2.5, UART Mode Control Register (UARTCR): Clarification added. The NEF field is read/write Section 56.4.2.5, UART Mode Control Register (UARTCR): Valid configuration values added to RSE (Reduced Over Sampling Enable) field description. Section 56.4.2.5, UART Mode Control Register (UARTCR): Clarification added to DTU (Disable Timeout in UART mode) field description Section 56.4.2.5, UART Mode Control Register (UARTCR): Clarification added to WLS (Special Word Length in UART mode) field description Section 56.4.2.5, UART Mode Control Register (UARTCR): – Correction to WL1 field description for value 01: ““Parity sent is Even” deleted from “8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1Parity sent is Even” – Correction to WL0 field description for value 01: ““Parity sent is Even” deleted from “8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1Parity sent is Even” Section 56.4.2.6, UART Mode Status Register (UARTSR): Clarification added to NF field description. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8. Figure 1291 (Block diagram): Correction: “Rxbysy” changed to “Rxbusy” Updated Section 56.4.2.24, UART Current Timeout Register (UARTCTO) Updated Section 56.2.2, UART mode features

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter LINFlexD (continued) Updated Section 56.3.2.8, Baud rate generation Updated Figure 1291 (Block diagram) Updated Section 56.2.2, UART mode features Updated Section 56.3.2.5.1, Header error Updated Section 56.3.2.5.2, Bit error Updated Section 56.4.2.5, UART Mode Control Register (UARTCR) Updated Section 56.4.2.24, UART Current Timeout Register (UARTCTO) Updated Table 1292 (Interface signals) Updated Section 56.3.2.8, Baud rate generation Updated Section 56.3.2.8, Baud rate generation Updated description of MIS, CSP, ROSE, DTU, SBUR and OSR fields in Section 56.4.2.5, UART Mode Control Register (UARTCR) Changed the reset value of IOBE bit from 0 to 1 in Section 56.4.2.13, LIN Control Register 2 (LINCRR2) Updated description of CSP and OSR fields in Section 56.4.2.5, UART Mode Control Register (UARTCR) Added note to Section 56.4.2.12, LIN Checksum Field Register (LINCRR) Added note to Section 56.4.2.24, UART Current Timeout Register (UARTCTO) Updated Section 56.4.3, Register map Added note to PE bit in Section 56.4.2.6, UART Mode Status Register (UARTSR) Removed note in Section 56.4.2.12, LIN Checksum Field Register (LINCRR) Updated DTU bit description in Table 1268 (UARTCR field descriptions) Changed FRR to FBRR in Figure 1291 (Block diagram) and the note below the figure Updated description of PE bit in Section 56.4.2.6, UART Mode Status Register (UARTSR) Updated Section 56.2.1, LIN mode features Updated Section 56.2.2, UART mode features Updated "Noise" subsection in Section 56.3.2.5.6, Timeout error Updated note in Section 56.3.2.8, Baud rate generation Added note in Section 56.4.2.10, LIN Fractional Baud Rate Register (LINFRR) Added note in Section 56.4.2.11, LIN Integer Baud Rate Register (LINIBRR) Changed the reset value from "0" to "1" in Section 56.4.2.1, LIN Control Register 1 (LINCRR1) Section 56.4.2.5, UART Mode Control Register (UARTCR): Added note in MIS bit description Changed "RFC0" bit to "RFC" in the register diagram Updated WLS bit description Updated TDFL/TFC bit description Updated WL1 bit description Updated WLO bit description Section 56.4.2.6, UART Mode Status Register (UARTSR): Added note to OCF bit description Updated DRF/RFE bit description Added note under Table 1250 (BDRM access in UART mode) Updated RDFBM bit description in Section 56.4.2.22, Global Control Register (GCR) Updated Section 56.3.1.3, Break Note added in Section 56.3.4.7, UART transmitter Note added in Section 56.3.5.10, Use cases and limitations

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter LINFlexD (continued) Table 1292 (Interface signals): Added the following LIN signals- in_ps_clr_DRF in_ps_clr_DTF in_ps_clr_PE[3:0] in_ps_clr_FEF in_ps_clr_BOF in_ps_clr_TOF out_ps_clr_DRF out_ps_clr_DTF out_ps_clr_PE[3:0] out_ps_clr_FEF out_ps_clr_BOF out_ps_clr_TOF out_bdr32b_rx[31:0] Section 56.4.1, Memory map: Added a note Section 56.4.2.22, Global Control Register (GCR): Replaced the address offset from "4C+4×no_of_filters" with "4C+4×no_of_filters×slave" Table 1268 (UARTCR field descriptions): Replaced "Bit 12 is reserved and not implemented" with "Bit 19 is reserved and not implemented" in RDFL bit Section 56.3.2.5.6, Timeout error: added subheadings: "Response timeout mechanism" and "Header timeout mechanism". Section 56.3.2.5.7, Noise: content reorganized into new section subheading replaced Figure 13 (Start detection and sampling for over sampling rate = 4 with six separate figures from Figure 1303 (Start detection and sampling for over sampling rate = 4 (Case 1)) to Figure 1308 (Start detection and sampling for over sampling rate = 4 (Case 5)) Table 1292 (Interface signals): Replaced out_ps_clr_DRF with out_ps_DRF Replace out_ps_clr_DTF with out_ps_DTF Replace out_ps_clr_PE[3:0] with out_ps_PE[3:0] Replace out_ps_clr_FEF with out_ps_FEF Replace out_ps_clr_BOF with out_ps_BOF Replace out_ps_clr_TOF with out_ps_TOF Converted content tagged "PSI5-S_yes" to "Baud_rate_25=yes" Added footnotes in Table 1292 (Interface signals) for all PSI5-S UART interface signals. Added Section 56.3.2.7.1, Start detection mechanism Added Section 56.3.2.7.2, Delimiter detection mechanism Table 1292 (Interface signals): – Added input signals in_ps_clr_RMB, in_ps_bit_remap, in_ps_sleep, and in_ps_init – Added output signals out_ps_RMB, and out_ps_mode[1:0] Updated PCE, and DTU bit description in Section 56.4.2.5, UART Mode Control Register (UARTCR) Added note to SLEEP, and INIT bits in Section 56.4.2.1, LIN Control Register 1 (LINC1)

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Reset Generation Module (MC_RGM) Figure 1393 ('Functional' Event Short Sequence Register (RGM_FESS)): - Register bit 30 renamed from SS_ESR0 to SS_ESR1 Figure 1388 ('Destructive' Bidirectional Reset Enable Register (RGM_DBRE)): - Reset value changed to 0x0000_0000 (was 0x0000_8400) Section 57.3.1.13, Peripheral Reset Register 0 (RGM_PRST0): - SIUL_RST field added Table 1294 (MC_RGM memory map), Section 57.3.1.6, 'Functional' Event Reset Disable Register (RGM_FERD), Section 57.3.1.7, 'Functional' Event Alternate Request Register (RGM_FEAR), Section 57.4.3, External reset - RGM_FEAR[AR_ESR0] changed to Read Only (always reads 0) - RGM_FERD[D_ESR0] changed to Read Only (always reads 0) Table 1294 (MC_RGM memory map): - SIUL_RST R/W added to RGM_PRST0[16] at 0x610 Chapter Boot Assist Flash (BAF) Section 58.3.4, Optionally remap calibration RAM: - Correction: Initialization status register of the buddy device is BD_SIUL2_ISR (was BD_SIU_ISR) - Updated Section 58.3.3, Flow of control to include instructions for operation X and Y - Updated Figure 1422 (Flow of control during serial boot). The decision block "Life Cycle "In Field" removed because the main flowchart (Figure 1414 (Flow of control)) already checks this lifecycle status. - Minor editorial changes - Updated the following sections: Section 58.3.2, Device initialization Section 58.3.6.2, Security watchdog configuration Section 58.3.6.2.4, Security Watchdog CPU select (SEC_SWT_CPU) Section 58.1, Introduction Section 58.3.7, Search boot header and boot options Chapter System Status and Configuration Module (SSCM) - In "Serial boot from SCI" section, changed Boot Assist ROM (BAR) to Boot Assist Module (BAM). Chapter Power Management Controller digital interface (PMC_dig) Removed incorrect description that the RES bits could not be updated if they had been loaded from the Flash (DCF) as a destructive reset. - Removed LVD9 RES registers and references (other than LVDIF) - updated reset of REE_VD9_EBI to be 0 - updated reset of FEE_VD0_IF & IF_2 to be 0 - Added the StandbyRAM logic description into the document - For REE_15 and FEE_15 registers and DEVICE2, corrected reset values for bit16 from 1 to 0.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Power Management Controller digital interface (PMC_dig) (continued) – Corrected TRIM_ADJ_UNDER and TIM_ADJ_OVER in CTL_TD register to show 4-bit fields, not 5-bit fields. – Corrected CTL_TD register informations to say “included two four bit fields” rather than “includes two five bit fields”. – updated reset value EBI RES, FEE, REE registers – updated reset value EBI RES, FEE, REE registers – Removed LVD10_a RES regisiter – adding new VSIO_IF2 bit – adding description of 2nd Temp Sensor output to HSM – updating conditional text for TRIM register reset values to match updated Analog values – updated Temperature Sensor section to refer to four bit User adjust bits instead of 5 bit User adjust registers – added clarification of LVD/HVD bypass via TESTPIN – Updated TEMP_0,1,2,3 names to be consistent – Fixed wording of VD9 REE description to remove the reference to DCF loading – Added VREG1P2_CTRL register and description of VREG1P2_ENB bit – Fixed wording of GR_P HVD entries to say “high voltage detect” instead of “low” – Updated GR_P with removal of VD7F and adding of VD8F in same location – Removed HVD7_F from EPR_VD7, REE_VD7, RES_VD7, and FEE_VD7 – Added HVD8_F to EPR_VD8, REE_VD8, RES_VD8, and FEE_VD8 – Described new PORs in reset section. – Fixed the reset value of the TM_CTRL and VREG1P2_ENB in the TEST_CTRL reg and memory map – Updated the IE register description to replace IE7_F with IE8_F – Updated section 1.14 to replace HVD140_F with HVD145_F Other Modules Interfaces section. Changed LVD names. – LVD114_H renamed to LVD108_C – LVD114_F renamed to LVD108_F – LVD114_P renamed to LVD108_P – LVD114_C renamed to LVD112_C – LVD280_C renamed to LVD270_C – LVD280_F renamed to LVD270_F – LVD280_IF renamed to LVD270_IF – LVD280_IJ renamed to LVD270_IJ – LVD280_IM renamed to LVD270_IM – LVD300_A renamed to LVD295_A – LVD300_F renamed to LVD295_F Memory Map/Register Definition section. Changed LVD names. – LVD114 renamed to LVD108 – LVD280 renamed to LVD270 – LVD300 renamed to LVD295

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Power Control Unit (MC_PCU) Editorial changes Chapter Mode Entry Module (MC_ME) Table 1410 (Mode Configuration Registers (ME_<mode>_MC) Field Descriptions): – Added footnote to XOSCON bit regarding loss of oscillator clock. Table 1406 (Interrupt Status Register (ME_IS) Field Descriptions): – I_ICONF_CC: "ME_CCTLn" changed to "ME_CADDRn" in bit value descriptions Section 62.3.2.27, Core Control Register 0 (ME_CCTL0): – Following sentence deleted: "A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted." Section 62.3.2.28, Core Control Register (ME_CCTL1): – Following sentence deleted: "A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted." Section 62.3.2.29, Core Control Register (ME_CCTL2): – Following sentence deleted: "A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted." Section 62.3.2.30, Core Control Register 3 (ME_CCTL3): – Following sentence deleted: "A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted." Section 62.3.2.31, Core Control Register (ME_CCTL4): Following sentence deleted: "A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted." Deleted CADDR2 (not used on this chip) Chapter e200z420n3 Core Debug Support In Section 63.5.3, JTAG/OnCE pins, changed the sentence "Asserting dbg_dbgrq will cause the chip to exit the low power mode enabled by the setting of MSR_{WE}, as well as temporarily exiting the Waiting, Stopped or Halted power management states." to "Asserting dbg_dbgrq will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states." Chapter e200z425n3 Core Debug Support In Section 64.5.3, JTAG/OnCE pins, changed the sentence "Asserting dbg_dbgrq will cause the chip to exit the low power mode enabled by the setting of MSR_{WE}, as well as temporarily exiting the Waiting, Stopped or Halted power management states." to "Asserting dbg_dbgrq will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states."

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Debug and Calibration Interface (DCI) Added a note to provide more details about simultaneous usage of SW debug & HW debug modes Changed references to non-secure mode to SoC Debug_Enable signal in: – Section 65.1.3.4, Nexus reset control – Section 65.1.3.11, Security Added Table 1488 (Nexus reset truth table) Section 65.1.3.4, Nexus reset control: – Following note added: If Debug_Enable is de-asserted by SOC security modules (HSM or PASS), then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets Figure 1586 (DCI operating modes state diagram) updated to '1st Timeout + 1' in DCI_LFAST_SAFE -> DCI_JTAG transition Chapter JTAG Controller (JTAGC) Removed arrow from TDI to TAP instruction decoder in Figure 1596 (JTAG (IEEE 1149.1) block diagram). Chapter IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG) In Section 67.3, Introduction, changed "Operation with both 4-pin and 2-pin star scan topologies and a 4-pin series scan topology" to "Operation with both 4-pin and 2-pin scan topologies". Renamed Section 67.4, CJTAG implementation summary (was "TAP.7 controller classes) and removed all content except that previously contained in "CJTAG implementation summary". Deleted the "Scan Selection Directive (SSD)" section. Deleted content pertaining to the SSCAN function in Section 67.7.1, Overview, and Section 67.7.8, SP Format Chapter JTAG Data Communication (JDC) Editorial changes Chapter Sequence Processing Unit (SPU) Editorial changes Chapter JTAG Master (JTAGM) Editorial changes

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter DigRF - High Speed Debug Table 1587 (LCR field description) "LCR field description" - Added text "The internal loopback is not intended for board level functionality, so this feature should not be implemented." to the end of the field description of LCR[LVLPEN]. Section 71.6.2.10, Unsolicited Tx Control Register (UNSTCR) Updated field name for UNSTCR[25:31] to UNSHDR Section 71.7.6.2.2, Unsolicited frame receive flow - Updated UNSRSR[USXDV] to UNSRSR[URXDV]. - Updated RISR[TXUNSF] to RISR[RXUNSF], bullet 3. Section 71.6.2.17, PLL and LVDS Status Register (PLLLSR) Table 1595 (PLLLSR field descriptions), 'PLLSR field descriptions'. Swapped fields LDSLPS and LRSLPS. Section 71.6.2.9, LVDS Control Register (LCR) Updated reset value of the LVRXOP field to 100b. Reset value of LCR is now 0000_5024h. Chapter Development Tool Semaphore (DTS) Editorial changes Chapter Nexus Aurora Router (NAR) Bit 20 of NAR_CR must be written with 0 in Section 73.4.1.1, NAR control register (NAR_CR). Changed reset value of Section 73.4.1.2, NAR status register (NAR_ST) to 0x00000001 (QWA=1). Chapter e200z420n3 Nexus 3 Module - In Section 74.3, TCODEs supported: changed standard to IEEE-ISTO 5001-2003 In Section 74.2, Enabling Nexus 3 Operation, added sentence about Nexus3 enabling with EVTI on TRST_B negation or during Test-Logic-Reset TAP state. Added the following text to Section 74.4.13, Read/Write Access Control/Status (RWCS) register: "Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values." Added the following text to Section 74.17.1, Single write access procedure step 4, "The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer." Added nex_ahb_start pin to note following this procedure. Added the following text to Section 74.17.2, Block write access, procedure step 2, "The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer." Added the following text to Section 74.17.3, Single read access, procedure step 3, "The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer." Added nex_ahb_start pin to note following this procedure. Added the following text to Section 74.17.4, Block read access, procedure step 2, "The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer."

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter e200z420n3 Nexus 3 Module (continued) In Table 1651 (DS field descriptions), updated LPS field description In Section 74.2.1, Interaction with Low Power Modes, replaced the paragraph with: "The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as nex_clk remains active. In the Stopped state, nex_clk is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming nex_clk is active." Chapter e200z425n3 Nexus 3 Module – In Section 75.3, TCODEs supported: changed standard to IEEE-ISTO 5001-2003 In Section 75.2, Enabling Nexus 3 Operation, added sentence about Nexus3 enabling with EVTI on TRST_B negation or during Test-Logic-Reset TAP state. Added the following text to Section 75.4.13, Read/Write Access Control/Status (RWCS) register: "Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values." Added the following text to Section 75.17.1, Single write access procedure step 4, "The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer." Added nex_ahb_start pin to note following this procedure. Added the following text to Section 75.17.2, Block write access, procedure step 2, "The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer." Added the following text to Section 75.17.3, Single read access, procedure step 3, "The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer." Added nex_ahb_start pin to note following this procedure. Added the following text to Section 75.17.4, Block read access, procedure step 2, "The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer." In Table 1704 (DS field descriptions), updated LPS field description Section 75.2.1, Interaction with Low Power Modes, replaced the paragraph with: "The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as nex_clk remains active. In the Stopped state, nex_clk is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming nex_clk is active."

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter GTM Development Interface (GTMDI) Changed SSEL1/2 fields in GTMDI_ATOM_WPC1 register description figure. It was read only, the correct is read/write for these fields. Added variable for PIN value of PIN field of DID register and added to variable table. Changed conditional text for SPEA/B SSEL field selection. The current implementation only allows SPE0 selection for SPEA and SPE1 selection for SPEB. This is for backwards compatibility with previous version which had just one possible SPE selection for SPEA.B (SPE1/2 at in previous version). Chapter Nexus Crossbar Multi-Master Client (NXMC) Changed WPHIT packet type to variable in Supported public TCODE table Table 1810 (DC2 register field description): Added descriptive phrase to DC2[PME0] field: "... for the In-Circuit Trace (ICT) feature" Chapter Cyclic Redundancy Check Unit (CRC) Corrected the INV field from "read only" to read/write in Section 78.5.1.1, CRC Configuration Register (CRC_CFG) Corrected CRC-32 polynomial Equation 90. Deleted "and limitations" from section title Section 78.7, Use cases. Changed "IPS" to "peripheral bus" in Section 78.2.1, Standard features, Section 78.4.1, Peripheral bus interface, Table 1960 (CRC top level diagram) and Section 78.6, Functional description. Moved footnote in Table 1820 (CRC unit memory map) because it appeared to say N-squared. Replace CRC_CNTX_NUM with N. Added row to Table 1820 (CRC unit memory map) for CRC_OUTP_CHK. Changed INV bit in Section 78.5.1.1, CRC Configuration Register (CRC_CFG) from RO to R/W. Section 78.5.1.1, CRC Configuration Register (CRC_CFG): Added field "swap_bitwise" to bit 27 and field "swap_bytewise" to bit 26 Section 78.5.1.1, CRC Configuration Register (CRC_CFG): Swapped the bit description of bit 26 and bit 27 Chapter Memory Error Management Unit (MEMU) In Table 1825 (MEMU memory map), changed the reset values of the SYS_RAM_CERR_STS, PERIPH_RAM_CERR_STS, and FLASH_CERR_STS registers (were 0000_00FF, are 0000_0000). In Figure 1975 (System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDRn)), added the missing offset (as present in Table 1825 (MEMU memory map)). In Section 79.6.2.2, Error flag register (ERR_FLAG), deleted the duplicate paragraph ("Individual flags... cleared by SW").

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Indirect Memory Access (IMA) Section 80.3.4.5, IMA access: – Added IMA_STATUS register name to bits for clarification – Correction: READ_LOCK needs to be cleared for reads (not READ_UNLOCK). Changed “31 READ” to “31 EN” in Table 1849 (IMA_ENABLE field descriptions). Section 80.3.4.5.1, Read access: – Clarification: Step 1 is to perform the required consecutive writes to the READ_UNLOCK register Section 80.3.4.5.2, Write access: – Clarification: Step 1 is to perform the required consecutive writes to the READ_UNLOCK register Section 80.3.3.2, IMA Enable Access Register (IMA_ENABLE): – Note added: Back to back writes to the enable bit is not valid. It will still generate a two clock enable pulse and will be considered as a single access initiation. Section 80.3.4.4, IMA lock – New section Chapter Fault Collection and Control Unit (FCCU) -Added OD field in Section 81.6.3, FCCU Configuration Register (FCCU_CFG) register. -Changed “WKUP module” to “WKPU” in Section 81.5.6, NMI interface - Updated Section 81.5.9, EOUT interface - Updated the bit description of “FCCU_SET_CLEAR” bit in Section 81.6.3, FCCU Configuration Register (FCCU_CFG) - Section 81.5.9, EOUT interface: Removed the Note section. - Updated Table 1862 (FCCU error pin behavior on STATE transitions) Updated PS field description in Section 81.6.3, FCCU Configuration Register (FCCU_CFG) Applied Saf1129 in OD field description in Section 81.6.3, FCCU Configuration Register (FCCU_CFG) Updated ESTAT field description in Section 81.6.13, FCCU Status Register (FCCU_STAT) Updated DEBUG field descriptions in Section 81.6.1, FCCU Control Register (FCCU_CTRL) Section 81.9.1, Misconfigurations : Added a new point, “If the user programs the FCCU for a higher timeout value than what is hardcoded for FOSU, then FCCU reacts later than what FOSU would be expecting and as a result, FOSU will not generate destructive reset request on its timeout”. Updated Table 1905 (FCCU register map) with all the new bits added in registers. Replaced all instances of ‘safe clock’ with ‘IRCOSC clock’. Table 1862 (FCCU error pin behavior on STATE transitions): – Updated table name to “FCCU error pin behavior on STATE transitions” from “FCCU error pin behavior”. – “FAULT state” updated to read “Transition from NORMAL to FAULT state”. – “ALARM state” updated to read “Transition from NORMAL to ALARM state”. – “Fault disabled/enabled” updated to read “with Fault disabled/enabled”. Added note 2 in Section 81.6.28, FCCU EOUT Signaling Enable Registers (FCCU_EOUT_SIG_EN[0:3]) Added note in Section 81.6.12, FCCU IO Control Register (FCCU_EINOUT)

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Self-Test Control Unit (STCU2) Removed section “Glossary, Acronyms & Abbreviations” Section 82.7.1, Off-Line Self-Test Sequence: deleted instance of “when pll_lock is high” Section 82.7.2, On-Line Self-Test Sequence: deleted instance of “also when pll_lock is high” Section 82.4.8.3, Write access time-out: replaced “that after AUTOLOCK_VALUE number of clock pulses lock” with “that, after 1024 STCU2 clock cycles, lock” Table 1908, STCU_RUNSW field descriptions: replaced instances of “pll_lock” with “PLL lock signal” Section 82.7.2, On-Line Self-Test Sequence: replaced instance of “pll_lock monitor” with “PLL lock signal monitor” Section 82.4.5, PLL interface: Replaced “As soon as it happens, forces the switch with the PLL output clock” with “As soon as the lock happens, the STCU forces the clock to switch to the PLL output clock” Chapter Register Protection (REG_PROT) Editorial changes Section 83.3.2.4, Global Configuration Register (GCR): – UAA bit moved to bit 8 Section 83.1.1, Overview: – Following note added: Please see the device configuration chapter of the device reference manual for details of the protected registers and bytes. Section 83.3.2.4, Global Configuration Register (GCR): – Intro text changed to: “This register controls the configuration of the locking of the SLB's and the level of user access allowed to the protected registers” Section 83.1.3, Modes of operation: – Following sentence removed: “For further details about the availability please see the chapter of the corresponding module” Chapter Password module (PASS) Section 84.3.1.4, Challenge Input register (CINn): – Following paragraph deleted: “To protect against voltage manipulations, complete 256-bit password must be considered a valid password challenge. An illegal value results in a failed challenge, so the password must be chosen appropriately.” Section 84.3.1.6.2, Hardware Interlock (HI): – “Once the hardware pin is <i>negated</i>, programming of BAF is controlled by PASS_LOCK0[BAF] bits” changed to “Once the hardware pin is <i>asserted high</i>....” Section 84.3.1.11, Password Group n - Lock 3 status register (LOCK3_PGn): Figure 2114 (LOCK3_PGn Register): – Note added to RL field bits: “Not all devices include this field. See the device reference manual's chip configuration chapter for details.” Section 84.5.4, Censoring/uncensoring the device: – Example, including “Censoring the device” table, deleted – Fixed conditional text error that was causing content to run into the heading of the next section.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Password module (PASS) (continued) Section 84.5.5, Unlocking the debug interface via JTAG: – Following text deleted: “The password needs to be set in UTest, and the same rules apply as to other passwords. The result of setting an invalid JTAG password is the debug interface cannot be unlocked via the JTAG password challenge register.” Chapter Tamper Detect Module (TDM) Minor editorial changes Chapter ASC Modeling Unit Section 86.1, Introduction: – Correction: The [AMU] block consists of a local 32 KB RAM with 64-bit ECC. (was incorrectly stated as 50 KB RAM. Chapter Wakeup Unit (WKPU) Table 2012 (NSR field descriptions): repaired unresolved cross-reference to footnote “Index ‘x’ designates ... and 2 is the IOP.” Renamed Section 87.5.2, Non-maskable interrupt initialization (was “Non-Maskable Interrupts”) Numerous editorial changes. Chapter GTM103 Integration (GTMINT) Module Table 2014 (High level memory map): – Moved: FIFO0_MEMORY moved from 0x0001A000 to 0x00019000 – Moved: MAP moved from 0x00000040 to 0x00000F00 – Moved: MCFG moved from 0x00000080 to 0x00000F40 – Moved: MCS1 moved from 0x00040000 to 0x00031000 – Moved: MCS2 moved from 0x00050000 to 0x00032000 – Moved: MCS3 moved from 0x00060000 to 0x00033000 – Moved: MCS4 moved from 0x00070000 to 0x00034000 – Moved: MCS5 moved from 0x00080000 to 0x00035000 – Moved: MCS6 moved from 0x00088000 to 0x00036000 – Moved: MCS0_MEMORY moved from 0x00038000 to 0x00038000 – Moved: MCS1_MEMORY moved from 0x00048000 to 0x00040000 – Moved: MCS2_MEMORY moved from 0x00058000 to 0x00048000 – Moved: MCS3_MEMORY moved from 0x00068000 to 0x00050000 – Moved: MCS4_MEMORY moved from 0x00078000 to 0x00058000 – Moved: MCS5_MEMORY moved from 0x00088000 to 0x00060000 – Moved: MCS6_MEMORY moved from 0x00098000 to 0x00068000 – New: GTM-IP AEI Bridge Registers at 0x00000030 – New: DPLL REGION EXT at 0x00028E00

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter GTM103 Integration (GTMINT) Module (continued) Table 2015 (GTMINT Module Memory Map): GTM-IP TOP-Level Configuration Registers: – New: GTM-IP (GTM_EIRQ_EN) at 0x00020 TIM AUX Registers (new): – GTM-IP TIM[m] auxiliary source selection register (m = 0..3) (GTM_TIM[m]_AUX_IN_SRC) at 0x00040+m*0x04 TIM0 Input Mapping Module (MAP) Registers: – Moved: MAP Control register (MAP_CTRL) moved from 0x00040 to 0x00F00 Memory Configuration (MCFG) Configuration Registers: – Moved: Memory layout configuration (MCFG_CTRL) moved from 0x00080 to 0x00F40 Output Compare Unit (CMP) Configuration Registers: – New: IRQ mode configuration register (CMP_EIRQ_EN) at 0x00214 Clock Management Unit (CMU) Configuration Registers: – New: Global clock control denominator (CMU_FXCLK_CTRL) at 0x00344 Broadcast Module (BRC) Configuration Registers (n = 0...11): – New: Software reset (BRC_EIRQ_EN) at 0x00474 Interrupt Concentrator Module (ICM) Configuration Registers: – New: ICM Interrupt group register for module error interrupt information (ICM_IRQG_MEI) at 0x00630 – New: ICM Interrupt group register 0 for FIFO channel error interrupt information (ICM_IRQG_CEI0) at 0x00634 – New: ICM Interrupt group register 1 for TIM0..3 channel error interrupt information (ICM_IRQG_CEI1) at 0x00638 – New: ICM Interrupt group register 3 for MCS0..3 channel error interrupt information (ICM_IRQG_CEI3) at 0x00640 Sensor Pattern Evaluation (SPE) Registers (AAi = i*0x80, i = 0..1): – New: SPE Interrupt enable register (SPE[i]_EIRQ_EN) at 0x0083C + AAi – New: SPE Interrupt generation by software (SPE[i]_REV_CNT) at 0x00840 + AAi – New: IRQ mode configuration register (SPE[i]_REV_CMP) at 0x00844 + AAi Timer Input Module (TIM) Configuration Registers (BBi = i*0x800, n = 0...7, i = 0..3): – Moved: Channel n control (TIM[i]_CH[n]_CTRL) moved from 0x01000+n*0x80 + BBi to 0x01024+n*0x80 + BBi – Moved: Channel n filter parameter 0 (TIM[i]_CH[n]_FLT_RE) moved from 0x01004+n*0x80 + BBi to 0x0101C+n*0x80 + BBi – Moved: Channel n filter parameter 1 (TIM[i]_CH[n]_FLT_FE) moved from 0x01008+n*0x80 + BBi to 0x01020+n*0x80 + BBi – Moved: Channel n general purpose 0 (TIM[i]_CH[n]_GPR0) moved from 0x01010+n*0x80 + BBi to 0x01000+n*0x80 + BBi – Moved: Channel n general purpose 1 (TIM[i]_CH[n]_GPR1) moved from 0x01014+n*0x80 + BBi to 0x01004+n*0x80 + BBi – Moved: Channel n SMU counter (TIM[i]_CH[n]_CNT) moved from 0x01018+n*0x80 + BBi to 0x01008+n*0x80 + BBi – Moved: Channel n SMU shadow counter (TIM[i]_CH[n]_CNTS) moved from 0x0101C+n*0x80 + BBi to 0x01010+n*0x80 + BBi

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter GTM103 Integration (GTMINT) Module (continued) Table 2015 (GTMINT Module Memory Map): Timer Input Module (TIM) Configuration Registers (BBi = i*0x800, n = 0...7, i = 0..3)(cont): – Moved: Channel n interrupt notification (TIM[i]_CH[n]_IRQ_NOTIFY) moved from 0x01020+n*0x80 + BBi to 0x0102C+n*0x80 + BBi – Moved: Channel n interrupt enable (TIM[i]_CH[n]_IRQ_EN) moved from 0x01024+n*0x80 + BBi to 0x01030+n*0x80 + BBi – Moved: Channel n software interrupt force (TIM[i]_CH[n]_IRQ_FORCINT) moved from 0x01028+n*0x80 + BBi to 0x01034+n*0x80 + BBi – Moved: Channel n IRQ mode configuration register (TIM[i]_CH[n]_IRQ_MODE) moved from 0x0102C +n*0x80 + BBi to 0x01038+n*0x80 + BBi – Moved: TIM Global software reset (TIM[i]_RST) moved from 0x1400 + BBi to 0x107C + BBi – New: Channel n control (TIM[i]_CH[n]_ECTRL) at 0x01028+n*0x80 + BBi – New: Channel n TDU control (TIM[i]_CH[n]_TDUC) at 0x01014+n*0x80 + BBi – New: Channel n TDU control (TIM[i]_CH[n]_TDUV) at 0x01018+n*0x80 + BBi – New: Channel n SMU counter (TIM[i]_CH[n]_ECNT) at 0x0100C+n*0x80 + BBi – New: Channel n software interrupt force (TIM[i]_CH[n]_EIRQ_EN) at 0x0103C+n*0x80 + BBi – New: Channel n IRQ mode configuration register (TIM[i]_IN_SRC) at 0x01078 + BBi – Deleted: Channel n TDU control (TIM[i]_CH[n]_TDU) at 0x0100C+n*0x80 + BBi Timer Output Module (TOM) Configuration Registers (CCi = i*0x800, i = 0...2, n = 0...15): – Moved: TGC0 global control reg (TOM[i]_TGC0_GLB_CTRL) moved from 0x08000 + CCi to 0x08030 + CCi – Moved: TGC0 enable/disable control reg (TOM[i]_TGC0_ENDIS_CTRL) moved from 0x08004 + CCi to 0x08070 + CCi – Moved: TGC0 enable/disable status reg (TOM[i]_TGC0_ENDIS_STAT) moved from 0x08008 + CCi to 0x08074 + CCi – Moved: TGC0 action time base register (TOM[i]_TGC0_ACT_TB) moved from 0x0800C + CCi to 0x08034 + CCi – Moved: TGC0 output enable control reg (TOM[i]_TGC0_OUTEN_CTRL) moved from 0x08010 + CCi to 0x08078 + CCi – Moved: TGC0 output enable status reg (TOM[i]_TGC0_OUTEN_STAT) moved from 0x08014 + CCi to 0x0807C + CCi – Moved: TGC0 force update control reg (TOM[i]_TGC0_FUPD_CTRL) moved from 0x08018 + CCi to 0x08038 + CCi – Moved: TGC0 internal trigger control (TOM[i]_TGC0_INT_TRIG) moved from 0x0801C + CCi to 0x0803C + CCi – Moved: TGC1 global control register (TOM[i]_TGC1_GLB_CTRL) moved from x08040 + CCi to x08230 + CCi – Moved: TGC1 enable/disable control reg (TOM[i]_TGC1_ENDIS_CTRL) moved from x08044 + CCi to 0x08270 + CCi – Moved: TGC1 enable/disable status reg (TOM[i]_TGC1_ENDIS_STAT) moved from x08048 + CCi to 0x08274 + CCi – Moved: TGC1 action time base register (TOM[i]_TGC1_ACT_TB) moved from x0804C + CCi to 0x08234 + CCi

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter GTM103 Integration (GTMINT) Module (continued) Table 2015 (GTMINT Module Memory Map): Timer Output Module (TOM) Configuration Registers (CCi = i*0x800, i = 0...2, n = 0...15)(cont): – Moved: TGC1 output enable control reg (TOM[i]_TGC1_OUTEN_CTRL) moved from 0x08050 + CCi to 0x08278 + CCi – Moved: TGC1 output enable status reg (TOM[i]_TGC1_OUTEN_STAT) moved from 0x08054 + CCi to 0x0827C + CCi – Moved: TGC1 force update control reg (TOM[i]_TGC1_FUPD_CTRL) moved from 0x08058 + CCi to 0x08238 + CCi – Moved: TGC1 internal trigger control (TOM[i]_TGC1_INT_TRIG) moved from 0x0805C + CCi to 0x0823C + CCi – Moved: TOM Channel m control register (TOM[i]_CH[n]_CTRL), i=0...2, n=0...15 moved from 0x08080 + n*0x0040 + CCi to 0x08000 + n*0x0040 + CCi – Moved: TOM Channel n CCU0 counter register (TOM[i]_CH[n]_CN0) moved from 0x08084 + n*0x0040 + CCi to 0x08014 + n*0x0040 + CCi – Moved: TOM Channel n CCU0 compare register (TOM[i]_CH[n]_CM0) moved from 0x08088 + n*0x0040 + CCi to 0x0800C + n*0x0040 + CCi – Moved: TOM Channel n CCU0 compare shadow register (TOM[i]_CH[n]_SR0) moved from 0x0808C + n*0x0040 + CCi to 0x08004 + n*0x0040 + CCi – Moved: TOM Channel n CCU1 compare register (TOM[i]_CH[n]_CM1) moved from 0x08090 + n*0x0040 + CCi to 0x08010 + n*0x0040 + CCi – Moved: TOM Channel n CCU1 compare shadow register (TOM[i]_CH[n]_SR1) moved from 0x08094 + n*0x0040 + CCi to 0x08008 + n*0x0040 + CCi – Moved: TOM channel n status (TOM[i]_CH[n]_STAT) moved from 0x08098 + n*0x0040 + CCi to 0x08018 + n*0x0040 + CCi – Moved: TOM channel n interrupt notification register (TOM[i]_CH[n]_IRQ_NOTIFY) moved from 0x0809C + n*0x0040 + CCi to 0x0801C + n*0x0040 + CCi – Moved: TOM channel n interrupt enable register (TOM[i]_CH[n]_IRQ_EN) moved from 0x080A0 + n*0x0040 + CCi to 0x08020 + n*0x0040 + CCi – Moved: TOM channel n software interrupt generation (TOM[i]_CH[n]_IRQ_FORCINT) moved from 0x080A4 + n*0x0040 + CCi to 0x08024 + n*0x0040 + CCi – Moved: Channel n IRQ mode configuration register (TOM[i]_CH[n]_IRQ_MODE) moved from 0x080A8 + n*0x0040 + CCi to 0x08028 + n*0x0040 + CCi

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter GTM103 Integration (GTMINT) Module (continued) Table 2015 (GTMINT Module Memory Map): ARU-connected Timer Output Module (ATOM) Registers (DDi = i*0x800, i=0..4, m=0..7): – Moved: AGC Global control register (ATOM[i]_AGC_GLB_CTRL) moved from 0x0D000 + DDi to 0x0D040 + DDi – Moved: AGC0 Enable/disable control register (ATOM[i]_AGC_ENDIS_CTRL) moved from 0x0D004 + DDi to 0x0D044 + DDi – Moved: AGC Enable/disable status register (ATOM[i]_AGC_ENDIS_STAT) moved from 0x0D008 + DDi to 0x0D048 + DDi – Moved: AGC Action time base register (ATOM[i]_AGC_ACT_TB) moved from 0x0D00C + DDi to 0x0D04C + DDi – Moved: AGC Output enable control register (ATOM[i]_AGC_OUTEN_CTRL) moved from 0x0D010 + DDi to 0x0D050 + DDi – Moved: AGC Output enable status register (ATOM[i]_AGC_OUTEN_STAT) moved from 0x0D014 + DDi to 0x0D054 + DDi – Moved: AGC Force update control register (ATOM[i]_AGC_FUPD_CTRL) moved from 0x0D018 + DDi to 0x0D058 + DDi – Moved: AGC Internal trigger control register (ATOM[i]_AGC_INT_TRIG) moved from 0x0D01C + DDi to 0x0D05C + DDi – Moved: ATOM Channel m control register (ATOM[i]_CH[m]_CTRL) moved from 0x0D080 + m*0x0080 + DDi to 0x0D004 + m*0x0080 + DDi – Moved: ATOM Channel m status register (ATOM[i]_CH[m]_STAT) moved from 0x0D084 + m*0x0080 + DDi to 0x0D01C + m*0x0080 + DDi – Moved: ATOM Channel m ARU read address register (ATOM[i]_CH[m]_RDADDR) moved from 0x0D088 + m*0x0080 + DDi to 0x0D000 + m*0x0080 + DDi – Moved: ATOM Channel m CCU0 counter register (ATOM[i]_CH[m]_CN0) moved from 0x0D08C + m*0x0080 + DDi to 0x0D018 + m*0x0080 + DDi – Moved: ATOM Channel m CCU0 compare register (ATOM[i]_CH[m]_CM0) moved from 0x0D090 + m*0x0080 + DDi to 0x0D010 + m*0x0080 + DDi – Moved: ATOM Channel m CCU0 compare shadow register (ATOM[i]_CH[m]_SR0) moved from 0x0D094 + m*0x0080 + DDi to 0x0D008 + m*0x0080 + DDi – Moved: ATOM Channel m CCU1 compare register (ATOM[i]_CH[m]_CM1) moved from 0x0D098 + m*0x0080 + DDi to 0x0D014 + m*0x0080 + DDi – Moved: ATOM Channel m CCU1 compare shadow register (ATOM[i]_CH[m]_SR1) moved from 0x0D09C + m*0x0080 + DDi to 0x0D00C + m*0x0080 + DDi – Moved: ATOM channel m interrupt notification register (ATOM[i]_CH[m]_IRQ_NOTIFY) moved from 0x0D0A0 + m*0x0080 + DDi to 0x0D020 + m*0x0080 + DDi – Moved: ATOM channel m interrupt enable register (ATOM[i]_CH[m]_IRQ_EN) moved from 0x0D0A4 + m*0x0080 + DDi to 0x0D024 + m*0x0080 + DDi – Moved: ATOM channel m software interrupt generation (ATOM[i]_CH[m]_IRQ_FORCINT) moved from 0x0D0A8 + m*0x0080 + DDi to 0x0D028 + m*0x0080 + DDi – Moved: Channel m IRQ mode configuration register (ATOM[i]_CH[m]_IRQ_MODE) moved from 0x0D0AC + m*0x0080 + DDi to 0x0D02C + m*0x0080 + DDi

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter GTM103 Integration (GTMINT) Module (continued) Table 2015 (GTMINT Module Memory Map): FIFO Configuration Registers (m=0...7, i=0): – Moved: FIFO Channel m write pointer register (FIFO[i]_CH[m]_WR_PTR) from 0x18500+m*0x20 to 0x1841C+m*0x40 + EEi – Moved: FIFO Channel m read pointer register (FIFO[i]_CH[m]_RD_PTR) from 0x18504+m*0x20 to 0x18420+m*0x40 + EEi – Moved: FIFO m interrupt notification register (FIFO[i]_CH[m]_IRQ_NOTIFY) from 0x18600+m*0x20 to 0x18424+m*0x40 + EEi – Moved: FIFO m interrupt enable register (FIFO[i]_CH[m]_IRQ_EN) from 0x18604+m*0x20 to 0x18428+m*0x40 + EEi – Moved: FIFO m register to force interrupt by software (FIFO[i]_CH[m]_IRQ_FORCINT) from 0x18608+m*0x20 to 0x1842C+m*0x40 + EEi – Moved: FIFO m IRQ mode control register (FIFO[i]_CH[m]_IRQ_MODE) from 0x1860C+m*0x20 to 0x18430+m*0x40 + EEi – New: FIFO m error interrupt enable register (FIFO[i]_CH[m]_EIRQ_EN) at 0x18434+m*0x40 + EEi Digital PLL Module (DPLL): – Moved: 9 bit ID information for input signal PMTR_n (8:0), n=0...23 (DPLL_ID_PMTR_n) moved from 0x28050 + n*0x0004 to 0x28100 + n*0x0004 – New: Error Interrupt enable register (DPLL_EIRQ_EN) at 0x28050 – New: Status Register (DPLL_STATUS) at 0x280FC – Moved: control bits for actions 0...3 (DPLL_ACB_0) from 0x281C0 to 0x28F00 – Moved: control bits for actions 4...7 (DPLL_ACB_1) from 0x281C4 to 0x28F04 – Moved: control bits for actions 8...11 (DPLL_ACB_2) from 0x281C8 to 0x28F08 – Moved: control bits for actions 12...15 (DPLL_ACB_3) from 0x281CC to 0x28F0C – Moved: control bits for actions 16...19 (DPLL_ACB_4) from 0x281D0 to 0x28F10 – Moved: control bits for actions 20...23 (DPLL_ACB_5) from 0x281D4 to 0x28F14 DPLL RAM 1B Map (Regions 1b and 1c): – New: control bits for actions 24...27 (DPLL_ACB_6) at 0x28F18 – New: control bits for actions 28...31 (DPLL_ACB_7) at 0x28F1C – Deleted: For ACTION_n in emergency mode (PDT_Sn), n=0...23 at 0x28560 + n*0x0004

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter GTM103 Integration (GTMINT) Module (continued) Table 2015 (GTMINT Module Memory Map): Multi Channel Sequencer (MCS) Configuration Registers (FFi = i*0x1000, i = 0..3, n = 0..7): – New: MCS Channel n error interrupt enable register (MCS[i]_CH[n]_EIRQ_EN) at 0x30054+n*0x80 + FFi – New: Reserved space at address range (0x3C000 – 0x3FFFC) + GGi – Moved: MCS Channel n control register (MCS[i]_CH[n]_CTRL) from 0x30000+n*0x80 + FFi to 0x30020+n*0x80 + FFi – Moved: MCS Channel n Program counter register (MCS[i]_CH[n]_PC) from 0x30004+n*0x80 + FFi to 0x30040+n*0x80 + FFi – Moved: MCS Channel n GPR registers (MCS[i]_CH[n]_R[y]), y=0...7 from 0x30008+n*0x80+ y*0x04 + FFi to 0x30000+n*0x80+ y*0x04 + FFi – Moved: MCS Channel n ACB register (MCS[i]_CH[n]_ACB) from 0x30028+n*0x80 + FFi to 0x30024+n*0x80 + FFi – Moved: MCS Channel n interrupt notification register (MCS[i]_CH[n]_IRQ_NOTIFY) from 0x3002C+x*0x80 + FFi to 0x30044+x*0x80 + FFi – Moved: MCS Channel n interrupt enable register (MCS[i]_CH[n]_IRQ_EN) from 0x30030+n*0x80 + FFi to 0x30048+n*0x80 + FF – Moved: MCS Channel n software interrupt generation register (MCS[i]_CH[n]_IRQ_FORCINT) from 0x30034+n*0x80 + FFi to 0x3004C +n*0x80 + FFi – Moved: IRQ mode configuration register (MCS[i]_CH[n]_IRQ_MODE) from 0x30038+n*0x80 + FFi to 0x30050+n*0x80 + FFi – Moved: MCS Control register (MCS[i]_CTRL) from 0x30400 + FFi to 0x30074 + FFi – Moved: MCS Clear trigger control register (MCS[i]_CTRГ) from 0x30404 + FFi to 0x30028 + FFi – Moved: MCS Set trigger control register (MCS[i]_STRG) from 0x30408 + FFi to 0x3002C + FFi – Moved: MCS Channel reset register (MCS[i]_RST) from 0x3040C + FFi to 0x30078 + FFi – Moved: MCS Error register (MCS[i]_ERR) from 0x30410 + FFi to 0x3007C + FFi Chapter Emulation Device Introduction Throughout: – “DigRF” is now “LFAST” Chapter Buddy Device Crossbar and RAM No changes Chapter Buddy Device-System Integration Unit Lite (BD_SIUL2) No changes

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Buddy Device-Debug and Calibration Interface (DCI) Throughout: – “DigRF” is now “LFAST” – Formatting and other editorial changes Section 92.1.3.3, Software enabled debug and calibration mode: – Note added recommending using either hardware-based debug or software-based debug, but not both. Section 92.1.3.4, Nexus reset control: – Changed references to non-secure mode to SoC Debug_Enable signal Section 92.1.3.11, Security: – Changed references to non-secure mode to SoC Debug_Enable signal – Added Figure 2035 (Nexus reset truth table) Figure 2172 (DCI operating modes state diagram) updated to ‘1st Timeout + 1’ in DCI_LFAST_SAFE -> DCI_JTAG transition Chapter Buddy Device-JTAG Controller (JTAGC) Section 93.1.2, Features: – Changed CENSOR_CTRL to JTAG_PASSWORD Section 93.3.4, JTAG_PASSWORD register: – Changed CENSOR_CTRL to JTAG_PASSWORD (was CENSOR_CTRL register) Section 93.3.4, JTAG_PASSWORD register: – Changed CENSOR_CTRL to JTAG_PASSWORD (was CENSOR_CTRL register) Section 93.4.4, JTAGC block instructions: – Changed CENSOR_CTRL to JTAG_PASSWORD Section 93.4.4.5, ENABLE_JTAG_PASSWORD instruction: – Changed CENSOR_CTRL to JTAG_PASSWORD (was ENABLE_CENSOR_CTRL instruction) Chapter Buddy Device - JTAG Master (JTAGM) Throughout: – “DigRF” is now “LFAST” Chapter Buddy Device-Nexus Aurora Router (NAR) Section 95.4.1.1, NAR control register (NAR_CR): Table 2058 (NAR_CR field descriptions): – Bits 20–16 must be written with 0. Section 95.4.1.2, NAR status register (NAR_ST): Figure 2205 (NAR status register (NAR_ST)): – Changed reset value to 0x00000001 (QWA=1). Section 95.4.7, Trace Memory fill level and partition configuration register (NAR_AHFPAR): Table 2069 (NAR_AHFPAR field description): – Bits 31–25 must be written with 0. – Bits 23–21 must be written with 0. – Bits 15–12 must be written with 0.

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter Buddy Device-Nexus Aurora Link (NAL) <i>Figure 2222 (NAL block diagram):</i> – UFC output removed from Flow Control block in Aurora Protocol Engine block <i>Section 96.1, Transmit operation:</i> – “After each packet transfer, the NAL adds...” changed to “After each payload transfer, the NAL adds...” – Note regarding clock compensation added <i>Section 96.4.2, NAL General Control Register (NAL_GCR):</i> <i>Figure 2224 (NAL General Control Register (NAL_GCR)):</i> – Removed ALCLBE and APELBE fields. These are internal test bits that should not be changed. <i>Table 2082 (NAL_GCR field descriptions):</i> – Removed ALCLBE and APELBE fields. These are internal test bits that should not be changed. <i>Section 96.4.3, NAL Training Control Register (NAL_TCR):</i> <i>Table 2083 (NAL_TCR field descriptions):</i> – Changed STE field 0 setting to Reserved. Chapter Buddy Device-Nexus Aurora PHY (NAP) <i>Section 97.1, Introduction:</i> – “and CRC checking” removed from following sentence: “...it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called Lane Alignment in Aurora terminology), 10b/8b decoding, decode and disparity error tracking and CRC checking in the received data stream.” <i>Section 97.1.2, Modes of operation:</i> – “The functional configuration is determined by the IO Port Control setting ...” changed to “The functional configuration is determined by the NAR_NAPCR setting...” <i>Section 97.2, External signal description:</i> – Clarification: Each of the NAP signals consists of a positive and negative differential pair. – Differential pair signal names added to NAP signals table Chapter DWPU <i>Section 98.3.1, Nexus Out Interface (NOI):</i> Detail added: – Messages received for filtering are from NAR – The NAR only initiates a transfer to the DWPU when it has full beats messages – Incoming messages are buffered – New figure: <i>Figure 2229 (Incoming message format)</i>

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter DWPU (continued) Section 98.3.3, MCA has the capability of providing one fully concatenated message to a CSPU each clock. Client Specific Processing (CSP): Detail added: – The Client Specific Processing (CSP) block receives the messages from MCA. The received message are processed after extracting all the relevant fields. Different types of message from various clients are received and processed differently based on their type by a specialized unit called the Client Specific Processing Unit (CSPU). – All message types other than data write messages are dropped Section 98.3.3.1.1, Standard data trace write message format: – Optional TSTAMP) field is 26 bits (was 32 bits) Section 98.3.4, Address Tag Check (ATC): – Section significantly rewritten with added detail Section 98.3.7, Nexus Client Interface (NCI): – NCI also supports the buffer reallocation to increase the Q size Section 98.3.8, IEEE 1149.1 (JTAG) Test Access Port: – New Section 98.3.9, Enabling DWPU operation: – New Section 98.3.10, Enabling TAP controller: – New Section 98.3.11, Register access via JTAG: – New Section 98.4, Configuration: – New Section 98.4.1, Tag RAM initialization: – New Section 98.5, Register definition: Table 2093 (JTAG DWPU registers): – Reset value for Global Control (GLB_CTRL) register changed to 0x0000_0002 (was 0x0000_0000) – Reset value for Out of Range Check Control 1 (ORC_CTRL1) register changed to 0x0000_0000 (was 0x0000_000F) – Reset value for Global Status (GLB_STAT) register changed to 0x0000_0000 (was 0x0004_0000) Register figures in register subsections: – Access level deleted—see Table 2093 (JTAG DWPU registers) for access – “Offset” replaced with “Index” in JTAG DWPU registers Section 98.5.1.2, Source Mapping Control n (SRC_MAP_CTLRn): Figure 2241 (SRC_MAP_CTLRn register): – Bit numbering changed. Least significant bit is bit 0. Table 2096 (SRC_MAP_CTLRn field descriptions): – CSP_TYPE0 field value list updated to reflect field's 3-bit width and include all values – CSP_TYPE1 field value list added

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter DWPU (continued) Section 98.5.1.3, Filter Address Range Start Address n (FAR_SA_CFGn): Figure 2242 (FAR_SA_CFGn register): – Bit numbering changed. Least significant bit is bit 0. Section 98.5.1.4, Filter Address Range Region Size n (FAR_RS_CFGn): Figure 2243 (FAR_RS_CFGn register): – Bit numbering changed. Least significant bit is bit 0. Section 98.5.1.5, Tag RAM Control (TRC_CTRL): Figure 2244 (TRC_CTRL register): – Bit numbering changed. Least significant bit is bit 0. – New fields added: ERR, EXC, PROG, INV, DSM – Reserved bits all read only 0 Table 2099 (TRC_CTRL field descriptions): – New fields added: ERR, EXC, PROG, INV, DSM – Note: ERR field is not present in the IPS mapped version of register. Section 98.5.1.6, Tag RAM Configuration Address (TRC_ADR): Figure 2245 (TRC_ADR register): – Bit numbering changed. Least significant bit is bit 0. Section 98.5.1.7, Tag RAM Configuration Data (TRC_DATA): Figure 2246 (TRC_DATA register): – Bit numbering changed. Least significant bit is bit 0. Section 98.5.1.8, Out of Range Check Control 1 (ORC_CTRL1): Figure 2247 (ORC_CTRL1 register): – Bit numbering changed. Least significant bit is bit 0. – Reserved bits all read only 0 Table 2102 (ORC_CTRL1 field descriptions): – TORn fields' "0" value definitions changed: Trigger is generated if data write value is within the range $ORC_CFGn_DL \leq data_wr_value \leq ORC_CFGn_DH$ ("$< ORC_CFGn_DH$" changed to "$\leq ORC_CFGn_DH$") – TORn fields' "1" value definitions changed: Trigger is generated if data write value is outside the range $data_wr_value < ORC_CFGn_DL$ or $data_wr_value > ORC_CFG0_DH$ ("or $data_wr_value \geq ORC_CFGn_DH$" changed to "or $data_wr_value > ORC_CFGn_DH$") Section 98.5.1.9, Out of Range Check Control 2 (ORC_CTRL2): Figure 2248 (ORC_CTRL2 register): – Bit numbering changed. Least significant bit is bit 0. – MODE field deleted Table 2103 (ORC_CTRL2 field descriptions): – MODE field deleted Section 98.5.1.10, Global Status (GLB_STAT): Table 2104 (GLB_STAT field descriptions): – Bit numbering changed. Least significant bit is bit 0. – DIDLE field added Table 2104 (GLB_STAT field descriptions): – DIDLE field added

Table 2152. Document revision history (continued)

Date	Revision	Changes
06-Jun-2013	4 (continued)	Chapter DWPU (continued) Section 98.5.1.11, Out of Range Configuration Address n (ORC_CFGn_ADR): Figure 2250 (ORC_CFGn_ADR register): – Bit numbering changed. Least significant bit is bit 0. Section 98.5.1.12, Out of Range Configuration Data High n (ORC_CFGn_DH): Figure 2251 (ORC_CFGn_DH register): – Bit numbering changed. Least significant bit is bit 0. Section 98.5.1.13, Out of Range Configuration Data Low n (ORC_CFGn_DL): Figure 2252 (ORC_CFGn_DL register): – Bit numbering changed. Least significant bit is bit 0. Section 98.5.1.14, Out of Range Configuration Data Mask n (ORC_CFGn_DM): Figure 2253 (ORC_CFGn_DM register): – Bit numbering changed. Least significant bit is bit 0. Chapter Buddy Device-Nexus Read Write Access (RWA) No changes Chapter Buddy Device-DigRF - High Speed Debug Throughout: – “DigRF” is now “LFAST” – Minor editorial changes Table 2129 “LCR field description”: – Added text “The internal loopback is not intended for board level functionality, so this feature should not be implemented.” to the end of the field description of LCR[LVLPEN]. Section 100.6.2.11, Unsolicited Tx Control Register (UNSTCR): – Updated field name for UNSTCR[25:31] to UNSHDR Section 100.7.3.2.3, Auto correlation: – Detail added regarding clock phases Section 100.7.6.2.2, Unsolicited frame receive flow: – Correction: At beginning of section, “If UNSRSR[USXDV] = 0” changed to “If UNSRSR[URXDV] = 0”. – Correction: In step 3 at beginning of section, “...and the RISR[TXUNSF] = 1...” changed to “...and the RISR[RXUNSF] = 1”.. Section 100.6.2.18, PLL and LVDS Status Register (PLLSR): – Table 2137, ‘PLLSR field descriptions’. Swapped fields LDSLPS and LRSPLS. Section 100.6.2.10, LVDS Control Register (LCR): – Updated reset value of the LVRXOP field to 100b. Reset value of LCR is now 0000_5024h. Section 100.10.3.1, Clocking module requirements for high speed phases: – Detail added on connections and routing.
04-Dec-2013	5	Updated Disclaimer.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER'S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR "AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL" INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2013 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com